UNIVERSIDAD CARLOS III DE MADRID

ESCUELA POLITÉCNICA SUPERIOR

INGENIERÍA DE TELECOMUNICACIÓN

PROYECTO FIN DE CARRERA

DETECCIÓN Y SEGUIMIENTO DE PERSONAS EN ENTORNOS INDOOR EMPLEANDO REDES DE SENSORES INALÁMBRICAS CON TECNOLOGÍA IMOTE2

AUTOR: Aurora Mourelle Aguado

TUTORES: Antonio Artés Rodríguez

Ángel Bravo Santos

14 de marzo de 2011

Título: *DETECCIÓN Y SEGUIMIENTO DE PERSONAS EN ENTORNOS INDOOR EMPLEANDO REDES DE SENSORES INALÁMBRICAS CON TECNOLOGÍA IMOTE2.*

Autor: *AURORA MOURELLE AGUADO*

Tutores: *ANTONIO ARTÉS RODRÍGUEZ*

ÁNGEL BRAVO SANTOS

La defensa del presente Proyecto Fin de Carrera se realizó el día 14 de marzo de 2011; siendo calificada por el siguiente tribunal:

Presidente: *Joaquín Míguez Arenas*

Secretario *Marcelino Lázaro Teja*

Vocal *Jaime José García Reinoso*

Habiendo obtenido la siguiente calificación:

Calificación:

Presidente **Secretario** **Vocal**

Agradecimientos

Después de tantos años de trabajo y esfuerzo, este proyecto pone fin a lo que hace tiempo empecé con tanta ilusión y que tanto esfuerzo me ha costado. Aprovecho estas líneas para dar las gracias a tanta gente que me ha acompañado durante todo este tiempo.

En primer lugar, a mi familia, en especial a mis padres y mi hermana. Gracias por apoyarme siempre, por estar ahí cuando más lo necesitaba y por enseñarme a no rendirme y a perseguir lo que quiero. Sin vosotros no sería quien soy.

Gracias a mis tutores, Antonio y Ángel, por haber confiado en mi para este proyecto, por su paciencia y por haberme permitido colaborar con el departamento trabajando en ello.

A mis amigos de siempre, Ale, Elena, Naiara y Antonio, que siempre han estado a mi lado, en los buenos momentos y sobre todo en los malos. Las buenas amistades son para toda la vida.

Una mención especial a mis amigos y compañeros durante la carrera: Carlos, Alberto, Mario, Javi, Raquel y Alicia, por hacer que tantos años de esfuerzo, tantas noches interminables y prácticas que nunca acababan hayan merecido la pena.

A David, mi antecesor en la beca, que me enseñó lo necesario para empezar a trabajar en el proyecto y a Jesse, que se queda ahora trabajando en ello, por su ayuda para hacer las pruebas con la cámara.

Especialmente quisiera agradecer a Isa, Raúl y a Camilo, por su gran colaboración durante las interminables pruebas con los sensores, pero sobre todo por haber sido mi mayor apoyo todo este tiempo.

Por último quisiera dar las gracias a mis compañeros del departamento: Isa, Blanca, Jorge, Luca, Raúl, Pablo, Wilton, Fran, Camilo, Katrin... que han hecho que mi estancia en el departamento haya sido de lo mejor que he vivido durante estos años.

Un hombre inteligente es aquel que sabe ser tan inteligente como para contratar gente más inteligente que él.

KENNEDY, John Fitzgerald

Resumen

En este trabajo se estudiará la viabilidad de un sistema de detección y seguimiento de personas en entornos *indoor* utilizando redes de sensores inalámbricas basadas en tecnología Imote2. El objetivo consiste en desarrollar una aplicación usando sensores pasivos para el seguimiento de personas mayores o con cierto grado de minusvalía que vivan solas, de forma que si sufren algún accidente, éste se pueda detectar y dar un aviso automáticamente a los servicios de emergencias.

Se realizará una descripción inicial de la plataforma empleada y de los sensores disponibles que permitirá determinar aquellos que resultarán útiles para la aplicación. Una vez seleccionados los sensores, se estudiará su funcionamiento mediante las calibraciones oportunas, realizando el procesado de datos necesario para determinar los futuros parámetros de funcionamiento de la aplicación.

Además de estudiar los sensores disponibles, se describirá la aplicación desarrollada que implementa el sistema de detección y seguimiento, basado en el uso de sensores de luz, de movimiento, acústicos y cámaras.

Finalmente, se resumirán los principales resultados y conclusiones extraídas y se proporcionarán una serie de sugerencias para continuar el proyecto en el futuro.

Palabras Clave: redes de sensores inalámbricas, Imote2, detección, seguimiento.

Abstract

This report will provide a feasibility study of an indoor people detection and tracking system using wireless sensor networks based on Imote2 technology. The objective is to develop an application using passive sensors to detect and track elderly people, or people with some physical handicap living alone, so that, if they suffer an accident, it can be automatically detected and an alarm given to the emergency services.

An initial description of the platform used and the available sensors will be provided, so that those sensors useful for the application can be selected. Once the sensors have been chosen, their performance will be studied by means of the pertinent calibrations and the required data processing to fix the parameters of the application will be described.

Besides studying the sensors in detail, the application developed implementing the detection and tracking system, based on light, movement, acoustic sensors and cameras, will also be described.

Finally, the main results and conclusions will be summarized, as well as a series of suggestions for future work.

Keywords: wireless sensor networks, Imote2, detection, tracking.

Índice general

Lista de Figuras

Lista de Tablas

Capítulo 1

Introducción

1.1. Motivación

Las redes de sensores inalámbricas son una de las tecnologías que más interés ha despertado en los últimos años. El carácter inalámbrico de las motas, junto con su reducido tamaño hacen que se planteen multitud de aplicaciones antes inviables. Por otra parte, un concepto que también despierta gran interés es el de domótica. Cada vez se tiende más hacia el diseño de casas inteligentes capaces, por ejemplo, de gestionar de forma eficiente la energía, detectar incendios, escapes de gas o controlar electrodomésticos de forma remota a través del teléfono móvil.

A partir de estas dos ideas, domótica y redes de sensores inalámbricas, se plantea la posibilidad de desarrollar aplicaciones que sirvan para ayudar a personas con minusvalías o personas mayores, que no sean totalmente dependientes, para que puedan vivir solas con la tranquilidad de que si algo les ocurre, se podrá avisar automáticamente a los servicios de emergencias.

En la actualidad existen cada vez más personas que se encuentran en la situación descrita y que corren el riesgo de sufrir un accidente sin que nadie sepa a tiempo que éste ha ocurrido. Según el informe anual 2008 de IMSERSO sobre personas mayores en España [14], a principios de 2007 el 16,7 % de la población en España tenía más de 65 años, lo que representa un aumento del 3,5 % en cinco años. Esta situación tiene consecuencias económicas por el aumento del gasto destinado a ayudas a este sector de población, y sociales debido a la asistencia que pueden necesitar. Con esta situación se han planteado desde hace algunos años soluciones basadas en nuevas tecnologías que permitan a estas personas mantener su autonomía sin perder la atención que requieren.

Las soluciones que se han dado hasta ahora requieren generalmente el uso de pulseras avisadoras u otros dispositivos que los individuos deben llevar consigo en todo momento, siendo

éste un problema por su incomodidad y por la posibilidad de que al usuario se le olvide llevarlo encima o no lo active si algo le ocurre. Esta situación, junto con el avance en la tecnología de redes inalámbricas, llevaron a plantearse la posibilidad de emplear un sistema totalmente pasivo capaz de realizar un seguimiento permanente de una persona en su casa y dar una alarma a los servicios de emergencia en caso de detectar una situación de peligro.

1.2. Visión General y Estado del Arte

1.2.1. Visión General

En los últimos años la detección y seguimiento de personas han sido objeto de gran interés, gracias sobre todo al creciente interés en desarrollar sistemas de vigilancia y seguridad en distintos ámbitos. Esta necesidad de seguridad ha llevado sobre todo a la instalación de sistemas basados en múltiples cámaras fijas que monitorizan todo lo que ocurre en un área determinada. El aumento del número de cámaras hace imposible que una persona sea capaz de controlarlas todas, de donde surge la necesidad de buscar sistemas automáticos de detección y seguimiento de personas, capaces de identificar las distintas actividades y de actuar en caso de que se detecte algo fuera de lo normal.

El problema de la detección y seguimiento de personas se puede plantear de la siguiente forma: dada un área de interés y un sistema de monitorización con uno o varios sensores, cómo se detecta la presencia de una persona y cómo se puede seguir su trayectoria. Se puede plantear este problema desde multitud de puntos de vista, dando lugar a infinitas soluciones. Puesto que no es un problema cerrado, lo primero que hay que plantearse a la hora de diseñar un sistema como el mencionado es para qué se va a utilizar. Si se quiere diseñar un sistema de vigilancia y seguridad, será necesario determinar qué actividades se definen como fuera de lo normal y el sistema tendrá que ser capaz de determinar automáticamente dichas actividades. En ese caso lo habitual es emplear imágenes proporcionadas por cámaras y un procesado de imagen adecuado. Si lo que se desea es un sistema de localización de personas en un entorno controlado, por ejemplo una residencia de ancianos, se podría emplear una solución autoportante, es decir, un sistema en el que las personas a las que se quiere seguir llevan encima un dispositivo, bien pasivo, bien activo, que se comunica con nodos externos proporcionando información para su localización. En este caso se podrían utilizar dispositivos RFID, infrarrojos o ultrasonidos fundamentalmente.

Una vez que se ha determinado la aplicación y el tipo de sensores que se van a emplear, es necesario encontrar la forma de emplear la información proporcionada por los sensores de

manera adecuada. Cuando sólo se utiliza un sensor, este proceso resulta relativamente sencillo, sin embargo, la fusión de información de diversos sensores que pueden ser además de distinto tipo, plantea un problema adicional.

1.2.2. Estado del Arte

Se han planteado hasta la fecha múltiples soluciones al problema del seguimiento de personas en entornos *indoor*, la gran mayoría proponen soluciones en las que el usuario debe llevar un dispositivo encima que se comunique con una infraestuctura de nodos estáticos.

El *Active Badge Location System* [24] propone una solución como la mencionada para localizar al personal dentro de una oficina. Este sistema se basa en que cada persona debe llevar una tarjeta de identificación que emite una señal infrarroja cada 15 segundos con un código único. La señal emitida es recogida por una red de sensores distribuidos por la oficina, y enviada a un nodo central que procesa los datos y presenta los resultados. El sistema requiere visión directa entre la tarjeta del usuario y los sensores.

La solución propuesta en [20] utiliza sensores de presión localizados bajo el suelo para detectar áreas de presencia y una solución autoportante basada en RFID para la identificación de usuarios. En [17] se recurre también a una solución autoportante, en este caso utilizando acelerómetros que permiten determinar el movimiento del usuario. La localización se realiza mediante nodos estáticos cuya posición es conocida y que se comunican con el nodo que lleva el paciente, de manera que su posición será la equivalente al nodo estático más cercano.

LocSense [15] es un sistema basado en la misma idea que los anteriores, utiliza nodos estáticos distribuidos en el edificio que se comunican con el dispositivo móvil que debe llevar el usuario. En este caso se utiliza el nivel de señal recibido por los nodos estáticos (RSSI, *Received Signal Strength Indicator*) como medida para la localización.

En cuanto a las soluciones al problema específico de dar asistencia remota a personas mayores o con problemas, la solución clásica es la teleasistencia domiciliaria. Se realiza a través de una línea telefónica específica, con un equipamiento de comunicaciones que permite al usuario establecer comunicación con el centro de asistencia desde cualquier punto del domicilio con sólo pulsar el botón del brazalete, reloj u otro elemento que debe llevar encima en todo momento.

La empresa Cystelcom [1] desarrolló un sistema de videocomunicación integrando comunicaciones IP y 3G orientado a la teleasistencia a personas mayores que permite al usuario ponerse en contacto con los servicios de asistencia a través del televisor. Además, permite la comunicación con los familiares conectados al sistema a través del ordenador o de un teléfono 3G. Este sistema

elimina la necesidad del brazalete, pero es necesario que el usuario se conecte a través de la televisión, por lo que si sufre un accidente en otro lugar el sistema no sería útil.

Todos los sistemas descritos presentan el mismo inconveniente: la necesidad de que la persona lleve encima algún tipo de dispositivo. Esto hace que no siempre resulten efectivos, ya que el usuario se puede olvidar de llevarlo. Un problema añadido a la teleasistencia clásica es que en muchas ocasiones el usuario no se encuentra físicamente capaz de activar el sistema de alarma.

El sistema SALVEO [2] creado por la empresa Pervaya es lo más parecido a la aplicación que se quiere desarrollar en este proyecto. En este sistema se utilizan sensores de movimiento distribuidos en el domicilio junto con sensores de temperatura para controlar el correcto funcionamiento de la calefacción y un sensor de contacto en la puerta principal para controlar las entradas/salidas. Toda la información recogida por los sensores se envía a una estación base que la analiza. Este sistema implementa un algoritmo de aprendizaje que permite determinar las rutinas de una persona y detectar situaciones anómalas basándose en variaciones bruscas de la rutina.

La diferencia entre el sistema SALVEO y la solución propuesta radica en la utilización de redes de sensores inalámbricas con una capacidad de cómputo que permite ejecutar los algoritmos necesarios en el propio nodo y decidir si es relevante transmitir la información, reduciendo así el tráfico y haciendo el sistema más eficiente desde el punto de vista energético. Además, el sistema presentado aquí no requiere de un aprendizaje previo para su funcionamiento

1.3. Objetivos

En este proyecto se plantea la posibilidad de desarrollar un sistema de detección y seguimiento de personas en entornos *indoor* con capacidad para realizar un análisis contextual que sirva para detectar situaciones de riesgo utilizando redes de sensores inalámbricas, en concreto, utilizando las motas Imote2 de Intel y las placas de sensores disponibles para dichas motas.

El sistema descrito es muy complejo y su completo desarrollo queda fuera del alcance de un Proyecto Fin de Carrera, así, los objetivos marcados son los siguientes:

- Realizar un análisis de la plataforma Imote2, las placas de sensores disponibles para la misma y seleccionar aquellos sensores que resultarán útiles para la implementación de la aplicación descrita.
- Estudiar el funcionamiento de los sensores seleccionados.
- Desarrollar un sistema de detección.

- Desarrollar un sistema de análisis contextual.
- Desarrollar una aplicación que integre detección y análisis contextual.

El desarrollo del sistema de seguimiento se dejará para futuras ampliaciones.

Así, el proyecto tiene dos partes bien diferenciadas: en primer lugar se realiza un estudio de la plataforma, los diferentes sensores disponibles y cómo se pueden utilizar para las funciones necesarias para desarrollar la aplicación, incluyendo las calibraciones realizadas para comprobar su funcionamiento. La segunda parte describe la aplicación desarrollada en la que se integran los diferentes tipos de sensores para resolver el problema expuesto.

1.4. Limitaciones

El problema planteado supone el desarrollo de una aplicación completa capaz de realizar detección, seguimiento y análisis contextual utilizando información de múltiples sensores de distintos tipos. Dada la complejidad del problema, este trabajo se centra fundamentalmente en la parte de detección, con un sistema de análisis contextual muy sencillo, de manera que puede servir de punto de partida para el desarrollo del sistema completo.

El desarrollo y diseño de una aplicación para una red de sensores incluye tareas de interconexión de redes, radio, procesamiento de señales, inteligencia artificial, administración de infraestructuras, optimización de recursos y algoritmos de gestión de potencia. Por tanto, la complejidad del trabajo no reside únicamente en la aplicación en sí, sino en los problemas a bajo nivel que es necesario resolver. Aquí se han utilizado en la medida de lo posible las soluciones ya existentes y que forman parte de la distribución de TinyOS (sistema operativo usado para redes de sensores), como la implementación del protocolo de acceso al medio (CSMA-CA). Otros problemas como el sincronismo no se han contemplado y se dejan para futuras ampliaciones.

Se asume que la aplicación se desarrolla para personas que viven solas, por lo que no se trata el problema de detección de múltiples objetivos.

1.5. Fases del Proyecto

Para el desarrollo del proyecto se han seguido las siguientes fases, algunas de las cuales se han ido realizando en paralelo como puede verse en el Diagrama de Gantt (Figura A.1), adjunto en el Apéndice A. A continuación se describe cada una de las fases.

1. **Documentación:** Se recabó información sobre las soluciones existentes tanto al problema de detección y seguimiento en general como al problema específico planteado (asistencia remota a personas mayores o con minusvalías). Además, se comenzó a estudiar la plataforma Imote2, el sistema operativo utilizado para redes de sensores (TinyOS) y el lenguaje de programación (nesC) necesario para programar las motas.

2. **Familiarización con la plataforma y su programación:** Antes de poder diseñar aplicaciones complejas, se probaron ejemplos sencillos de aplicaciones existentes con el objetivo de familiarizarse con el entorno de programación. Para ello se siguió un tutorial [12] facilitado en la página oficial de TinyOS, que cubre los aspectos básicos: desde la aplicación más sencilla hasta la comunicación entre motas y entre éstas y el ordenador.

3. **Desarrollo de aplicaciones para la calibración de los sensores utilizados:** Una fase importante del proyecto consiste en determinar el funcionamiento de los diferentes sensores que se van a utilizar, para ello se desarrollaron tantas aplicaciones como tipos de sensores y se realizaron las calibraciones oportunas. En esta fase también se programaron las funciones necesarias en Matlab para el procesado de los datos.

4. **Puesta en marcha de la placa IMB400:** Cuando se inició este proyecto no se disponía todavía de las placas multimedia (descritas en el Capítulo 3) y dado que se crearon hace pocos años, la versión de TinyOS no incluía (ni incluye hasta la fecha) interfaces para controlar los diferentes sensores de los que dispone. La Universidad de Stanford, a través de su contribución a TinyOS, proporcionó las interfaces necesarias para controlar la cámara, facilitando un ejemplo de aplicación. Esta interfaz, que no se ha modificado para este proyecto, no permite utilizar funcionalidades avanzadas de la cámara, simplemente permite tomar una imagen en dos formatos diferentes. El principal trabajo en esta fase estuvo relacionado con el sensor acústico, ya que no existía ningún trabajo al respecto. Inicialmente se trató de desarrollar un *driver* partiendo de una versión existente para un sistema operativo anterior, sin embargo surgieron numerosos problemas y finalmente se utilizó un driver facilitado en julio de 2010 a través del grupo de Yahoo asociado a Imote2 [3].

5. **Desarrollo de la aplicación:** Programación de la aplicación completa. Se puede subdividir en las siguientes fases:

   - Algoritmo de detección.
   - Algoritmo de análisis contextual.

- Desarrollo de un subsistema que integre detección y análisis contextual.
- Desarrollo de la aplicación final.

El desarrollo de la aplicación se fue realizando en paralelo a la puesta en marcha de la placa IMB400 y el desarrollo de las aplicaciones de calibración para los sensores.

Es importante recalcar que al iniciar el proyecto únicamente se tenían conocimientos sobre los aspectos básicos teóricos de redes de sensores inalámbricas. Tanto las características de TinyOS como del lenguaje de programación se aprendieron específicamente para el desarrollo del mismo. También es importante mencionar que cualquier duda o problema se tuvo que resolver a través del mencionado foro sobre Imote2 [3].

Finalmente destacar que se ha contribuido con este foro facilitando una variante de la aplicación para el control de la cámara desarrollada por la Universidad de Stanford y resolviendo dudas sobre algunos aspectos de la misma a otros participantes. La aplicación que se facilitó permitía el envío de una imagen tomada por un nodo inalámbrico a la estación base y de ésta al ordenador, siendo la transmisión inalámbrica la contribución realizada y que ha servido como referencia para otros usuarios del foro.

1.6. Estructura de la Memoria

Se ha realizado una introducción al problema planteado, la motivación para llevarlo a cabo, las soluciones existentes y los objetivos que se pretenden conseguir en este trabajo. El resto del documento está organizado de la siguiente forma: El Capítulo 2 da una breve introducción al sistema operativo TinyOS y al lenguaje de programación nesC. El Capítulo 3 describe la plataforma Imote2, detallando las características más importantes tanto del procesador como de las placas de sensores disponibles. En el Capítulo 4 se realiza un análisis detallado de los sensores que se van a utilizar en la aplicación, explicando cómo se van a utilizar y proporcionando los resultados de calibración obtenidos. Los Capítulos 2 y 3 describen la primera de las dos partes que componen el proyecto, el análisis detallado de la plataforma y en concreto de los sensores de interés. El Capítulo 5 describe en detalle la aplicación desarrollada, y en el Capítulo 6 se resumen los resultados y conclusiones extraídos.

Finalmente, se adjunta el presupuesto del proyecto y el Diagrama de Gantt del mismo, un ejemplo de fichero de salida y un resumen de los ficheros que componen la aplicación junto con una guía de utilización e instalación.

Capítulo 2

Introducción a TinyOS y nesC

En este capítulo se da una breve introducción al sistema operativo TinyOS, se explicarán sus características, su modelo de ejecución y las ventajas que presenta. También se describe el lenguaje de programación, nesC, en el que está escrito TinyOS y que se utiliza para la programación de las aplicaciones desarrolladas.

La relación entre TinyOS y nesC es muy estrecha, de manera que a pesar de comenzar describiendo TinyOS, muchos de los conceptos se entenderán mejor cuando se explique nesC.

2.1. TinyOS

TinyOS es un sistema operativo específicamente diseñado para sistemas embebidos, como las redes de sensores, que se creó con el fin de satisfacer las necesidades específicas de este tipo de sistemas, las cuales difieren sustancialmente de las necesidades cubiertas por un sistema operativo de propósito general. Los principales argumentos que justifican la necesidad de TinyOS son los siguientes:

1. Las limitaciones de capacidad de cómputo y de memoria de las motas comparados con un ordenador hacen necesario un sistema operativo ligero, sencillo y muy eficiente.

2. Al contrario que un ordenador, las motas ejecutan un único programa, por lo que en lugar de utilizar un sistema operativo de propósito general resulta más eficiente que éste se componga de una serie de componentes básicos que se unen para crear aplicaciones específicas.

3. El objetivo fundamental de las redes de sensores es la interacción con el entorno, obtener información sobre el mismo a través de distintos sensores y tomar decisiones en consecuen-

cia, por lo que las aplicaciones que corran sobre el sistema operativo estarán muy ligadas al hardware, y en concreto a los sensores disponibles.

4. La concurrencia es una característica clave. El sistema operativo debe ser capaz de manejar interrupciones generadas por la lectura de los sensores a la vez que la comunicación radio y las tareas definidas en la aplicación específica. El modelo de concurrencia utilizado en TinyOS es muy sencillo y se basa en tareas y eventos, tal y como se describe más adelante.

Una vez justificada la necesidad de TinyOS, se describen las características fundamentales de su diseño y funcionamiento.

Arquitectura basada en componentes: TinyOS está formado por una serie de componentes de sistema reutilizables. El concepto de *componente* es el de un elemento del sistema que proporciona un servicio concreto, como por ejemplo un temporizador, y que puede ser utilizado a alto nivel para diseñar una aplicación específica. Cada aplicación utiliza únicamente los componentes que necesite, conectándolos entre sí independientemente de su implementación. La mayoría de los componentes son módulos software, sin embargo algunos encapsulan hardware, como por ejemplo un componente para el uso de un sensor de luz. Esta diferencia es transparente a nivel de aplicación. La principal ventaja de este modelo es que los componentes que no se utilizan se pueden dejar fuera de la aplicación para optimizar los recursos.

Modelo de concurrencia basado en tareas y eventos: El modelo de concurrencia es muy sencillo y se basa en dos niveles de planificación:

- Eventos: Pensados para procesos cortos, como atender interrupciones. Pueden interrumpir la tarea o evento que se esté ejecutando.
- Tareas: Mecanismo para ejecutar procesos en segundo plano. Se ejecutan en su totalidad, sin poder interrumpir otras tareas. Los componentes pueden solicitar la ejecución de una tarea (*post*), lo cual ocurrirá cuando el planificador lo permita. El componente no espera a que se ejecute la tarea, vuelve inmediatamente después de la solicitud. El uso de tareas resulta útil cuando los requisitos temporales no son estrictos. En cualquier caso, para garantizar un retardo de ejecución bajo, las tareas deben ser cortas.

Con este modelo se permite que los eventos, que son sencillos y se ejecutan rápido, se

atiendan inmediatamente, pudiendo interrumpir la ejecución de las tareas, que son mucho más complejas.

Split-phase operations: Todos los procesos que impliquen un retardo alto, se realizan en dos fases; solicitud y señalización de que se ha completado a través de un evento. Un ejemplo típico es el envío de un paquete. Un componente invoca el comando *send* para enviar un mensaje radio, y el componente encargado de la comunicación radio señaliza con un evento *sendDone* que se ha completado la transmisión.

Todos los conceptos presentados se explicarán más detenidamente en el siguiente apartado, cuando se describa nesC y se proporcione un ejemplo de aplicación.

2.2. nesC

El lenguaje de programación nesC es una variante de C orientado a componentes y diseñado para programar aplicaciones para sistemas embebidos como las redes de sensores. Su creación está motivada para dar soporte al diseño de TinyOS, definiendo un modelo de concurrencia basada en eventos y una construcción de aplicaciones basada en componentes.

2.2.1. Componentes

Una aplicación nesC es un conjunto de componentes unidos entre sí. Un componente es un elemento que proporciona una funcionalidad específica y utiliza la funcionalidad proporcionada por otros. La implementación de un componente es transparente para los demás, de manera que el punto de acceso al mismo se realiza a través de una interfaz que oculta los detalles de su programación.

Interfaces

Las interfaces son bidireccionales: un componente la proporciona y otros la utilizan, de manera que cada interfaz declara una serie de funciones (*comandos*) que deben ser implementadas por el componente que proporciona la interfaz, y una serie de funciones (*eventos*) que los usuarios de la misma deben implementar. Si un componente quiere utilizar los comandos de una interfaz, necesariamente tiene que implementar los eventos definidos en la misma, aunque no los vaya a utilizar. Por otra parte, un mismo componente puede utilizar y proporcionar varias interfaces.

El siguiente código muestra parte de la interfaz *Timer*.

```
interface Timer{
  // Interfaz básica
  command void startPeriodic( uint32_t dt );
  command void startOneShot( uint32_t dt );
  command void stop();
  event void fired();
  // Resto de comandos omitidos por simplicidad
}
```

Si se quiere realizar una aplicación que utilice un temporizador, por ejemplo para tomar una lectura periódica de un sensor, será necesario llamar al comando *startPeriodic* de la interfaz *Timer*. La forma que tiene el componente que implementa la interfaz de indicar que se ha cumplido un ciclo es a través del evento *fired*, de manera que la aplicación diseñada deberá implementar qué ocurre cuando se señalice dicho evento, por ejemplo, encender o apagar un LED.

Módulos y Configuraciones

Existen dos tipos de componentes: módulos y configuraciones. Los módulos contienen el código propio del funcionamiento de la aplicación, y si el componente proporciona alguna interfaz, implementa los comandos de la misma. Las configuraciones se usan para unir componentes entre sí, conectando las interfaces usadas por unos a las interfaces proporcionadas por otros. A esta conexión se le denomina *wiring*.

Toda aplicación nesC está descrita por una configuración de alto nivel que une todos los componentes de la misma.

Para entender mejor el concepto de módulo y configuración, se proporciona a continuación el código de la aplicación básica Blink, que utiliza temporizadores para encender una serie de leds.

```
module BlinkC(){
  uses interface Timer<TMilli> as Timer0;
  uses interface Timer<TMilli> as Timer1;
  uses interface Timer<TMilli> as Timer2;
  uses interface Leds;
  uses interface Boot;
}
implementation{
  event void Boot.booted(){
    call Timer0.startPeriodic(250);
    call Timer1.startPeriodic(500);
    call Timer2.startPeriodic(1000);
  }
  event void Timer0.fired(){
    call Leds.led0Toggle();
  }
  event void Timer1.fired(){
    call Leds.led1Toggle();
  }
  event void Timer2.fired(){
    call Leds.led2Toggle();
  }
}
```

El módulo se declara con la palabra clave *module* seguida del nombre del componente y comienza con la declaración de las interfaces que se van a utilizar mediante la palabra clave **uses**. Destaca en este ejemplo el hecho de que se utilizan tres instancias de la interfaz *Timer*, lo cual es necesario si se quieren utilizar tres temporizadores distintos. La palabra clave **as** puede utilizarse para renombrar interfaces. La interfaz *Leds* permite encender y apagar los leds de la placa, y la interfaz *Boot* define el arranque de la aplicación, por lo que es necesario utilizarla en todas las aplicaciones.

Una vez declaradas las interfaces, la implementación es muy sencilla de entender. Comienza con la palabra clave **implementation** y a continuación se pasa a la programación en sí. En este caso únicamente se implementan los eventos definidos en las interfaces utilizadas. Al señalizarse el evento de arranque, se llama a las funciones proporcionadas por la interfaz *Timer* y éstas se ejecutan en modo *split-phase*, es decir, tras invocar la función, se espera a que se señalice su conclusión mediante un evento. Cuando esto ocurre, se ejecuta el código correspondiente, en este caso cuando salta una interrupción de uno de los *Timers*, se llama a la función que permite cambiar el estado de uno de los leds. Esta llamada ya no es *split-phase*, ya que su ejecución es muy corta.

En cuanto a la configuración, comienza con la palabra clave **configuration** seguida del nombre del archivo de configuración. Si se proporcionase alguna interfaz, ésta se declararía a continuación con la sintaxis **provides** *nomInterfaz*. En este caso esto no ocurre y se pasa directamente a unir las interfaces entre sí. Para ello es necesario declarar los componentes que proporcionan las interfaces utilizadas (cuatro primeras líneas de la implementación). Una vez hecho esto, se une la interfaz proporcionada por el componente a la utilizada en la aplicación. Por ejemplo, en el módulo se había declarado la interfaz Leds, y en la configuración se une a la interfaz Leds proporcionada por el componente LedsC mediante una flecha. El sentido de la flecha siempre será de usuario a proveedor.

```
configuration BlinkAppC{
}
implementation{
  components MainC, BlinkC, LedsC;
  components new TimerMilliC() as Timer0;
  components new TimerMilliC() as Timer1;
  components new TimerMilliC() as Timer2;
  BlinkC -> MainC.Boot;
  BlinkC.Timer0 -> Timer0;
  BlinkC.Timer1 -> Timer1;
  BlinkC.Timer2 -> Timer2;
  BlinkC.Leds -> LedsC;
}
```

Al realizar el *wiring*, si no hay ambigüedades en las interfaces proporcionadas por un componente, no es necesario especificar el nombre de la misma. Por ejemplo, si un componente proporciona únicamente una interfaz, como LedsC, no es necesario especificar que se va a unir la interfaz Leds usada en BlinkC a la interfaz Leds proporcionada por LedsC, puesto que se sobreentiende, y por tanto las siguientes líneas serían equivalentes:

```
BlinkC.Leds -> LedsC.Leds;
BlinkC.Leds -> LedsC;
BlinkC -> LedsC.Leds;
BlinkC -> LedsC;
```

2.2.2. Modelo de Ejecución

Los componentes se relacionan a través de comandos y eventos definidos en interfaces, sin embargo, es posible definir funciones privadas que no son accesibles a ningún otro componente. Estas funciones son idénticas a las que se puedan escribir en C, no llevan ningún modificador (*command* o *event*), pero pueden invocar comandos y señalizar eventos.

Todas las funciones y comandos vistos se ejecutan en el momento. Las tareas (*tasks*) permiten a una aplicación ejecutar un código en otro momento, proporcionando así un mecanismo para ejecutar tareas en segundo plano. Puesto que su ejecución no es inmediata, no es posible pasar parámetros a las tareas ni devolver resultados.

La declaración de cualquier tarea sigue la sintaxis:

```
task void nomtask();
```

Cuando se quiere hacer uso de este mecanismo se utiliza la sintaxis:

```
post nomtask();
```

2.3. Resumen

TinyOS y nesC permiten el desarrollo de aplicaciones para redes de sensores a través de un modelo basado en componentes que permite una gestión eficiente de los recursos.

Se ha dado una breve introducción a las principales características del entorno de programación, en [11] se da una descripción detallada del lenguaje nesC, profundizando en los conceptos

presentados aquí y explicando otros. Por otra parte, existe una sencilla guía de TinyOS [12] que explica la programación de aplicaciones para redes de sensores, desde la más sencilla como la presentada en este capítulo, hasta las que requieren comunicación entre motas y entre motas y el ordenador.

Capítulo 3

Descripción de la Plataforma

3.1. Imote2: Introducción

En los inicios de las redes de sensores inalámbricos y hasta hace poco, el principal objetivo de las investigaciones relativas al diseño de nodos inalámbricos se centraba en conseguir plataformas de muy bajo consumo que fuesen capaces de funcionar durante meses o incluso años. Esta limitación en el consumo combinada con la tecnología existente suponía que los nodos diseñados tuviesen una capacidad de procesamiento muy limitada, contando con microprocesadores de 8 bits, muy poca memoria y radios con codificaciones muy simples. Más adelante se empezaron a emplear microcontroladores de 16 bits y sistemas radio más sofisticados. Sin embargo, para aplicaciones en las que se necesitaba procesar los datos de forma más compleja, era necesario emplear una puerta de enlace, un nodo de mucha mayor potencia al que enviar todos los datos. El hecho de emplear un nodo especial para poder procesar la información tenía varios inconvenientes: el primero y fundamental es que dicho nodo resulta crítico para cualquier aplicación, el segundo es el desperdicio de ancho de banda para el envío de datos y el consumo extra de potencia que suponía el envío de dichos datos.

Para resolver estas limitaciones se pensó en diseñar nodos con una capacidad de cómputo mucho mayor, que fuesen capaces de procesar la información recogida por sus sensores de forma que la cantidad de datos intercambiados se redujese al mínimo. Los nodos Imote de Intel surgieron con dicho propósito y su evolución dio lugar a la segunda generación: Imote2.

En la Tabla 3.1 se pueden observar las diferencias entre las capacidades de distintos tipos de motas comerciales.

Tabla 3.1: Comparativa entre algunas motas comerciales.

Plataforma		Imote2	Mica2	MicaZ	TelosB	Tinynode
Fabricante		Crossbow	Crossbow	Crossbow	Crossbow	Shockfish
Procesador	Modelo	PXA271 Xscale	Atmega 128L	Atmega 128L	TI MSP430	TI MSP430
	Bits	32	8	8	16	16
	Frecuencia (MHz)	13 - 416	7.383	7.383	8	8
	Memoria (RAM, ROM)	32MB, 32MB	4kB, 128kB	4kB, 128kB	10kB, 48kB	48kB
	Interfaces	UART, SPI, I2C, I2S, AC97, Camera	UART, SPI, I2C	UART, SPI, I2C	UART, SPI, I2C	UART, SPI
	Corriente (activa, sleep)	31mA, 390uA	8mA, <15uA	8mA, <15uA	1.8mA, 5.1uA	1.8mA, 5.1uA
Radio	Modelo	CC2420	CC1000	CC2420	CC2420	XE1205
	Frecuencia (MHz)	2400	868	2400	2400	868
	Tasa (kbps)	250	76.4	250	250	1.2 ->152.3
	Standard	802.15.4		802.15.4	802.15.4	
	Potencia Tx (dBm)	-24 ->0	-20 ->5	-24 ->0	-24 ->0	0 ->12
	Sensibilidad (dBm)	-94	-98	-94	-94	-101
	Alcance exterior (m)	30	150	75 - 100	75 - 100	200 (76.8kbps)
	Corriente (max,min)	18.8mA, 1uA	27mA, <1uA	18.8mA, 1uA	18.8mA, 1uA	33mA, <1uA
Dimensiones (mm)		36x48x9	58x32x7	58x32x7	65x31x6	30x40

3.2. Placa del Procesador: IPR2400

Figura 3.1: Placa del procesador Imote2 (IPR2400) [5].

Imote2 es una plataforma avanzada de sensores inalámbricos diseñada para aplicaciones de redes de sensores con altos requisitos de procesamiento. Está construida en torno a un procesador Intel XScale PXA 271, e incluye una radio 802.15.4 con antena empotrada. Dispone de dos interfaces de sensores: la interfaz básica tiene como objetivo habilitar placas de sensores de bajo coste y soportar las interfaces de sensores más comunes. La interfaz avanzada expone las características más avanzadas del procesador, como pueden ser la interfaz para cámaras, los módulos de audio o el bus de alta velocidad. Ambas interfaces están compuestas por dos conectores cada una, cuyos pines (salvo el USB y el JTAG de la interfaz avanzada) se pueden programar para que funcionen como GPIO, dándole mayor flexibilidad a la placa.

La plataforma Imote2 tiene un diseño modular, de forma que permite apilar distintos módulos

de sensores para diseñar aplicaciones específicas. Además, cuenta con un módulo especial para el suministro de potencia. En la Figura 3.1 se puede ver el aspecto que presenta la placa del procesador, y la Figura 3.2 muestra esta misma placa con más detalle.

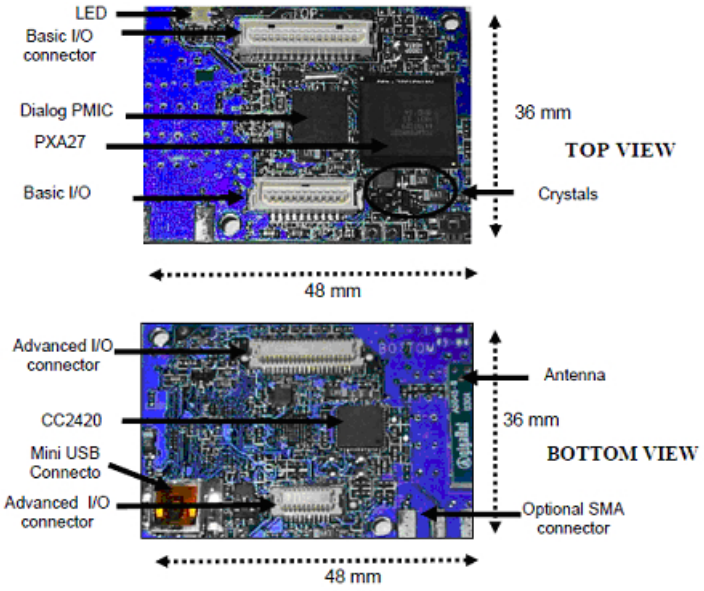


Figura 3.2: Detalle de la placa Imote2 (IPR2400) [5].

Las principales características hardware de la placa son las siguientes:

- Procesador PXA271 XScale con rango de frecuencias de 13 a 416 MHz
- Co-procesador MMX inalámbrico
- 256 kB SRAM, 32 MB FLASH, 32 MB SDRAM
- Radio 802.15.4 integrada y soporte para radios externas a través de SDIO y UART
- Antena 2.4 GHz integrada
- LED indicador de estado multicolor
- Conectores de expansión básico y avanzado con soporte para: 3xUART, I2C, 2xSPI, SDIO, I2S, AC97, host USB, Cámara I/F, GPIO
- Puerto Mini-USB para conexión directa con el PC
- Tamaño: 48 mm x 36 mm. Espesor de la PCB 1.75 mm

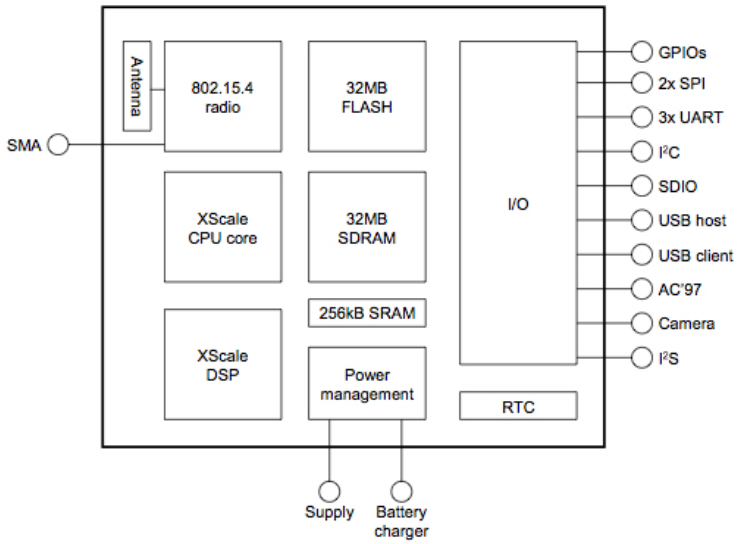


Figura 3.3: Diagrama de bloques de la placa Imote2 [10].

Uno de los componentes fundamentales de la placa es la radio CC2420, que implementa el standard Zigbee (802.15.4). Las características más importantes se resumen en la Tabla 3.2.

Tabla 3.2: Características de la radio CC2420.

Frecuencia	2.4 GHz
Tasa de tx.	250 kbps
Max. frame length	128 bytes
Tamaño buffer tx.	128 bytes
Tamaño buffer rx.	128 bytes
Modulación	O-QPSK y uso de DSSS

De acuerdo con el standard 802.15.4 [13], la máxima longitud de una trama a nivel físico es de 128 bytes, si se eliminan las cabeceras introducidas, que varían entre 13 y 25 bytes en función de si se utilizan direcciones origen/destino cortas o extendidas, el número de bytes de datos máximo que se pueden enviar en cada trama es de 103-115 bytes.

Dado que este proyecto no trata de analizar en detalle la plataforma que se va a emplear, sino que tiene un enfoque centrado en la aplicación, a continuación se analizan las placas de sensores disponibles para los nodos, de forma que se pueda determinar posteriormente qué tipo de soluciones se pueden plantear a la hora de implementar un sistema de detección y seguimiento de personas empleando este tipo de motas.

3.3. Placa de Sensores Básica: ITS400

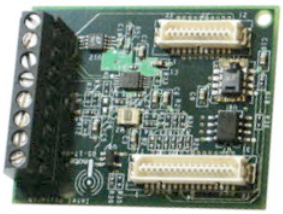


Figura 3.4: Placa de sensores básica ITS400 [6].

La placa ITS400 (Figura 3.4) combina una serie de sensores comúnmente empleados en aplicaciones de redes de sensores inalámbricos.

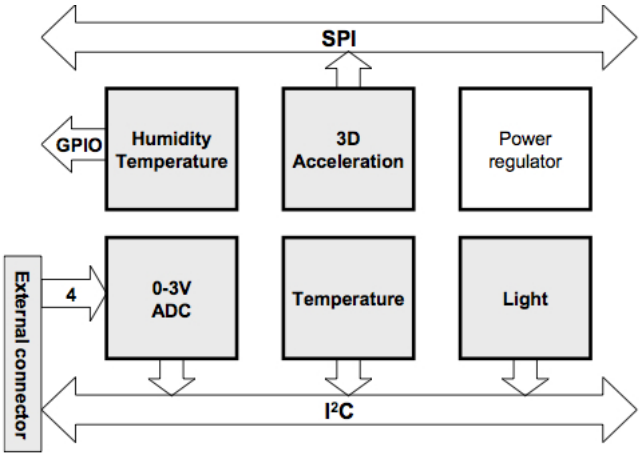


Figura 3.5: Diagrama de bloques de la placa de sensores básica ITS400 [6].

El acelerómetro de tres ejes es un ST Micro LIS3L02DQ que funciona en un rango de $\pm 2g$ y tiene una resolución de 12 bits. Los usos típicos especificados en las hojas de características son: Funciones activadas por movimiento para dispositivos móviles, sistemas antirrobo y navegación inercial, dispositivos de entrada para juegos y realidad virtual y monitorización y compensación de vibraciones.

El sensor de temperatura y humedad es el sensor de alta precisión Sensirion SHT15. De acuerdo con los datos del fabricante, proporciona una resolución típica en la medida de la humedad del 0.03 % de la humedad relativa (12 bits) con un rango del 0 al 100 % de la humedad relativa. En cuanto a la temperatura, la resolución típica es de 0.01 ℃(14 bits) con un rango de funcionamiento entre -40 ℃y 123.8 ℃. La resolución se puede variar modificando el registro de estado, pasando de 12 y 14 bits de resolución para humedad y temperatura respectivamente, a 8 y 12, lo que equivale a un 0.5 % de la humedad relativa y 0.04 ℃.

Además del sensor de temperatura anterior, la placa dispone de un sensor digital de temperatura TI TMP 175 que funciona entre -25 ℃y 85 ℃con una precisión de ±1,5 ℃y que permite la conexión de hasta 27 dispositivos en el bus, por lo que su aplicación fundamental es el control de temperatura en distintos dispositivos, como periféricos de ordenadores.

El sensor de luz TAOS TSL2651 dispone de dos canales conectados al procesador a través del bus I2C. El primer canal, de 16 bits, es sensible tanto a la luz visible como a la infrarroja, mientras que el segundo canal, también de 16 bits, es sensible únicamente a la luz infrarroja.

Por último, la placa dispone de un conversor analógico/digital (A/D) de propósito general, con cuatro canales y 12 bits de resolución.

3.4. Placa de Sensores Multimedia: IMB400

Figura 3.6: Placa multimedia IMB400 [4].

Tal y como su nombre indica, la placa IMB400 (Figura 3.6) añade funcionalidad multimedia a la plataforma Imote2, permitiendo la captura de imágenes, video y audio. Todos los datos se obtienen de forma digital para poder almacenarlos, transmitirlos o procesarlos en la placa principal. Además de la funcionalidad multimedia, dispone de un sensor de movimiento PIR (*Passive Infrared Sensor*).

Debido a las características de la placa, resulta perfecta para implementar sistemas de seguimiento de objetos y personas. La idea detrás del diseño del módulo multimedia es ofrecer una solución eficiente integrando cámara, audio y detección de movimiento en una misma plataforma. Cuando el sensor PIR detecta movimiento, puede activar la cámara para empezar a monitorizar en ese momento, ahorrando así energía.

La Figura 3.7 muestra el diagrama de bloques y las interfaces a través de las cuales se conecta a la placa del procesador.

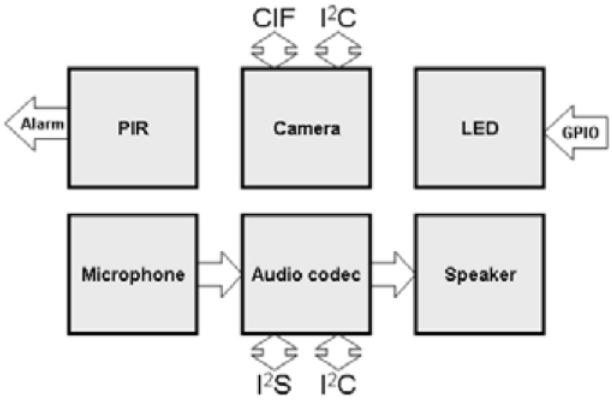


Figura 3.7: Diagrama de bloques de la placa multimedia IMB400 [4].

Los subsistemas de la plataforma constan de una cámara Omnivision OV7670 con resolución de hasta 640x480 pixels, con soporte para formatos RGB, YCbCr e YUV y una velocidad de captura de hasta 30 fps. Dispone de alguna funcionalidad de tratamiento de imagen integrada, como el escalado o el filtrado, además de ofrecer controles sobre la saturación, tono y contraste.

El subsistema de audio contiene un codec Wolfson WM8940, micrófono y altavoz, también con funcionalidad de procesado de señal integrado, como control automático de ganancia y varios filtros configurables .

El sensor de movimiento es el modelo standard de Panasonic AMN1, con un alcance máximo de 5 m y un ángulo de detección entre 80 y 100. El rango de velocidad de movimiento detectable es de 0.5 a 1.5 m/s. La detección más precisa se realiza hasta los 2 m, detectando movimientos mínimos de 30 cm.

3.5. Placa Interfaz: IIB2400

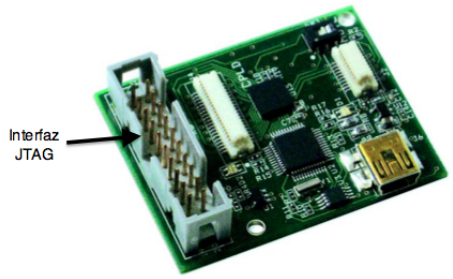


Figura 3.8: Placa interfaz IIB2400 [10].

Antes de pasar a describir en profundidad los sensores utilizados se explica el mecanismo utilizado para programar las motas. La placa IIB2400 (Figura 3.8) es el elemento esencial a

través del cual se realiza cualquier tipo de comunicación entre el ordenador y las motas.

Para cargar un programa en un nodo se necesitan los siguientes elementos:

- Placa interfaz IIB2400
- Programador arm-usb-jtag (Figura 3.9) + cable USB.
- Cable USB-miniUSB.

Figura 3.9: Programador JTAG.

El programador JTAG se conecta a la placa IIB2400 a través de la interfaz JTAG (señalada en la Figura 3.8). El otro extremo del programador se conecta mediante un cable USB al ordenador. Por otra parte, se debe conectar el cable usb-miniUSB entre el ordenador y la placa IIB2400. Una vez hecho esto, para programar una mota basta con conectar el procesador IPR2400 a la placa IIB2400 utilizando la interfaz avanzada.

Una vez se ha cargado el programa en la mota, se puede prescindir del programador, de manera que la comunicación entre el ordenador y la red de sensores se realiza utilizando la placa interfaz conectada a través del conector miniUSB al ordenador. Es importante mencionar que ambas placas, programador e interfaz, disponen de conector miniUSB. Para la comunicación con el ordenador la que debe estar conectada es la de la interfaz. El conector del procesador sirve únicamente para utilizar el ordenador como fuente de energía en lugar de las pilas.

3.6. Selección de Sensores

El principal objetivo del análisis de la plataforma era determinar aquellos sensores que resultarían útiles en la aplicación de interés.

Una vez analizados se concluye que de la placa básica se van a emplear únicamente los sensores de luz, detectando cambios de iluminación producidos por la sombra proyectada por el usuario sobre los sensores. Se pueden descartar los sensores de temperatura y humedad puesto

que resultan inútiles para detectar presencia. El acelerómetro a priori podría despertar interés en varios sentidos: podría resultar de gran utilidad si el usuario lo llevase encima, ya que registraría el movimiento y sería capaz de detectar una caída fácilmente, sin embargo, el objetivo es diseñar un sistema totalmente pasivo en el que el usuario no tenga que llevar nada consigo, por lo que esta forma de emplear los acelerómetros queda descartada. También se podría pensar en utilizarlos situados en el suelo, detectando las vibraciones que produce una persona al andar o cuando algo se cae al suelo. Se ha descartado esta posibilidad debido a la complejidad de posicionamiento que supondría para no interferir con la vida diaria, además, sería necesario desplegar una gran cantidad de sensores en cada zona, lo que aumentaría considerablemente el coste. Por último, se podrían utilizar como mecanismo para detectar aperturas y cierres de puertas, detectando así la entrada/salida de la persona en una determinada zona. Sin embargo, esto no resultaría eficiente por si mismo ya que puede ocurrir que el usuario entre en una zona sin necesidad de abrir la puerta o salga sin cerrarla. Para detectar la entrada en una zona se utilizarán los sensores de movimiento de la placa multimedia.

Efectivamente, los sensores de movimiento resultarán de especial utilidad para situar a la persona en una zona determinada de la casa, y servirán como mecanismo de ahorro de energía tal y como se explicará en el Capítulo 4. Por otra parte, los micrófonos y el códec de audio se podrían emplear como sistema de localización, sin embargo, implementarlo daría lugar a otro Proyecto Fin de Carrera completo, por lo que en este caso se van a emplear como parte del sistema de análisis contextual junto con la cámara, que resulta útil por razones evidentes. Pese a la potencial utilidad de la cámara, sólo se va a emplear en momentos muy concretos con el fin de ahorrar energía, ya que implementar algoritmos de procesado de imagen resultaría en un consumo excesivo que reduciría considerablemente la duración de las baterías.

En resumen, el sistema de detección se realizará empleando los sensores de movimiento y los de luz, mientras que el sistema de análisis contextual utilizará los sensores acústicos y las cámaras.

Capítulo 4

Descripción de los Sensores Utilizados

En este capítulo se describen los sensores empleados en la aplicación de detección y las calibraciones que han sido necesarias para determinar su funcionamiento y poder así fijar los diferentes parámetros de funcionamiento. Se han utilizado los sensores lumínicos y de movimiento para realizar la detección y el seguimiento de la persona, y los sensores acústicos y la cámara para el análisis contextual.

Como puede observarse, todos los sensores son pasivos, requisito indispensable de la aplicación ya que el objetivo es desarrollar una aplicación de bajo consumo e independiente del usuario.

4.1. Sensor de Movimiento

Los sensores de movimiento que se van a emplear son los incluidos en la placa IMB400, tal como se ha comentado en el Capítulo 3, estos sensores detectan movimiento a partir de la radiación infrarroja emitida por los objetos. Todos los objetos a una temperatura superior al cero absoluto emiten una radiación infrarroja cuya intensidad depende de la temperatura a la que se encuentre el objeto. Esta característica se puede aprovechar en un sistema de detección determinando un umbral que, al ser superado, signifique que se ha detectado la presencia de una persona.

El tipo más básico de sensor infrarrojo es el PIR (Passive Infra Red), un detector de movimiento basado en la detección de cambios en la intensidad de la radiación medida. Los sensores PIR funcionan en dos estados: el estado inactivo (o detección de movimiento), en el que el sensor está a la espera de un cambio en la radiación, y el estado activo, en el que se encuentra tras realizarse una detección.

Los sensores de movimiento van a constituir una parte fundamental de la aplicación, ya

que además de la funcionalidad de detección de personas que proporcionan, se emplean como mecanismo para el ahorro de energía de las motas. A continuación se proporciona una descripción más detallada de las características del modelo de sensor integrado en la placa IMB400, así como las calibraciones y pruebas realizadas para comprobar su funcionamiento.

4.1.1. Descripción General

Las principales características especificadas por el fabricante se detallan en la Tabla 4.1.

Tabla 4.1: Características PIR.

Detección	Alcance máximo	5 m
	Detección fina	Movimientos de 30cm a distancia máxima de 2 m
	Ángulo horizontal	100º
	Ángulo Vertical	82º
Eléctricas	Consumo Corriente	0.17 mA (típico), 0.3 mA (máx)
	Tiempo de estabilización	45 s

Generalmente este tipo de sensores se utilizan para detectar la presencia de personas en una determinada zona, de manera que se pueda, por ejemplo, realizar un control automático de la iluminación o de la refrigeración para así ahorrar energía.

A la hora de emplear los sensores PIR para una aplicación, es necesario tener en cuenta que, debido al principio de funcionamiento, bajo determinadas situaciones pueden no tener el funcionamiento esperado. Este comportamiento erróneo se puede dar de dos formas:

1. **Falsos positivos:** Detección de una fuente de radiación que no es una persona (asumiendo que es a una persona a quien se quiere detectar). Esto puede ocurrir en el caso de que un animal entre en la zona de detección, cuando el sensor está directamente expuesto a la luz solar, luz incandescente o, en general a cualquier fuente de infrarrojos lejanos. Además, se puede dar un falso positivo si se produce un cambio brusco en la temperatura ambiente, por ejemplo al encender el aire acondicionado o la calefacción.

2. **Dificultad para la detección:** La detección puede ser difícil si existe un objeto de cristal, acrílico o semejante entre el sensor y lo que se quiere detectar. Además, si la fuente de radiación (una persona, por ejemplo), no se mueve o lo hace muy rápido, también dificulta la detección.

El sensor de movimiento descrito anteriormente de forma genérica está integrado en la placa de sensores IMB400 tal y como se muestra en la Figura 3.7. No proporciona ninguna medida, simplemente genera una interrupción en caso de que detecte movimiento. Esta interrupción resulta útil como sistema de ahorro de energía, ya que permite tener la mota en modo de bajo consumo (*sleep mode*) hasta que se detecte la presencia de una persona.

4.1.2. Calibración

El objetivo de la calibración del sensor de movimiento es comprobar el rango de detección y el tipo de movimiento que es capaz de detectar. Para llevar a cabo dicha calibración, se colocó uno de los sensores a media altura en una esquina de un aula y se fue determinando, para cada ángulo, la distancia a la que el sensor detectaba movimiento. El mapa de detección se muestra en la Figura 4.1.

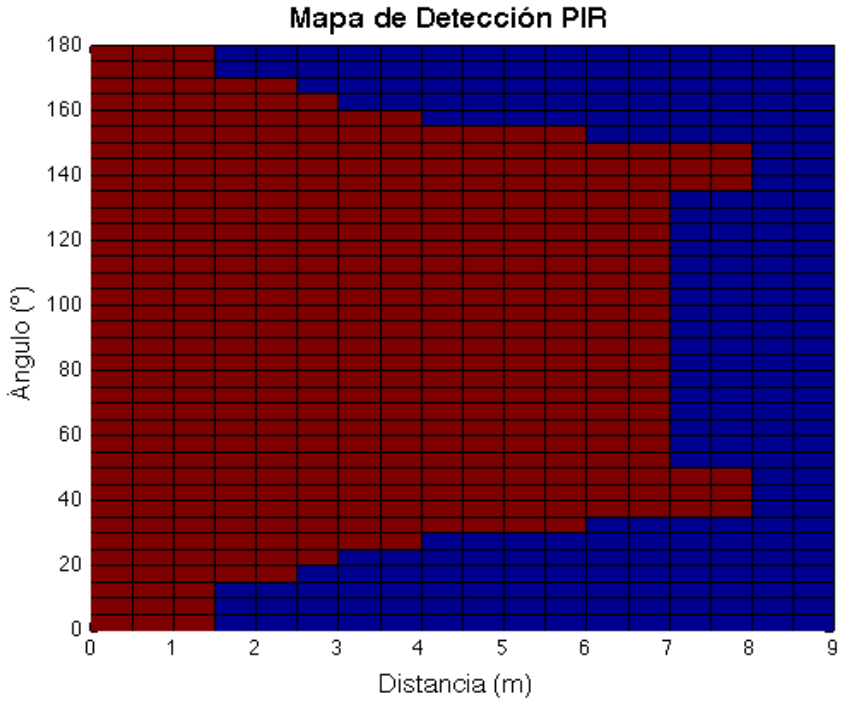


Figura 4.1: Mapa de detección del sensor PIR.

De acuerdo con los resultados obtenidos el alcance del sensor es mayor que el especificado por el fabricante. Es importante mencionar que al alejarse del sensor, el movimiento debe ser cada vez mayor para que sea detectado.

4.2. Sensores de Luz

Los sensores de luz TAOS TSL2561 integrados en la placa básica de sensores ITS400 de Crossbow para Imote2 van a ser el método principal de detección de la aplicación. Estos sensores están diseñados para medir la iluminación ambiente con el objetivo de aplicarlos en sistemas de regulación automática de la iluminación. Tal y como se ha mencionado en el capítulo anterior, disponen de dos canales de medida: uno de banda ancha, sensible tanto a la luz visible como a la infrarroja, y otro de banda estrecha, sensible únicamente a la luz infrarroja. En términos prácticos ambas medidas son proporcionales, por lo que el canal de banda estrecha, a priori, no resulta de especial utilidad.

La forma de emplear los sensores de luz en la aplicación consiste en detectar cambios bruscos de iluminación, de manera que, al comparar la medida con una serie de umbrales de detección, se pueda determinar si se ha detectado la presencia de una persona, o el apagado o encendido de luces en una habitación.

La elección de los umbrales de detección determinará el funcionamiento de la aplicación, ya que un umbral de detección demasiado bajo puede provocar falsos positivos y un umbral demasiado alto puede limitar demasiado el área de detección de un determinado sensor. Puesto que los umbrales van a resultar críticos, se han realizado una serie de calibraciones con el objetivo de determinar la respuesta del sensor, en diferentes condiciones de iluminación y ante la presencia de una persona a diferentes distancias y ángulos.

Es importante mencionar que las medidas reflejadas en todas las gráficas no responden a una unidad de iluminación, simplemente son el resultado de la lectura del sensor. Es posible obtener los resultados en lux aplicando un algoritmo que combina las medidas de los dos canales ponderadas adecuadamente [7], sin embargo, puesto que las unidades no son relevantes y el interés se centra exclusivamente en una comparación de umbrales, implementar el algoritmo de conversión resultaría en un aumento de la complejidad del código sin un aumento en la precisión de los resultados.

4.2.1. Calibración de los Sensores de Luz

Como se acaba de mencionar, el objetivo de la calibración es observar la respuesta de los sensores en diferentes condiciones de iluminación, en distintas posiciones y ante la presencia de un obstáculo (en el caso de interés una persona) a distintas distancias y ángulos.

Para poder obtener los datos se desarrolló una aplicación para Imote2 con una interfaz gráfica para poder controlar los sensores de forma remota sin necesidad de re-programarlos cada vez que se quiere variar un parámetro. La aplicación está pensada para poder utilizar varias motas simultáneamente y para tomar datos en distintos modos de manera que con la misma aplicación se puedan realizar diferentes tipos de pruebas. En la Figura 4.2 se muestra la interfaz gráfica para la aplicación.

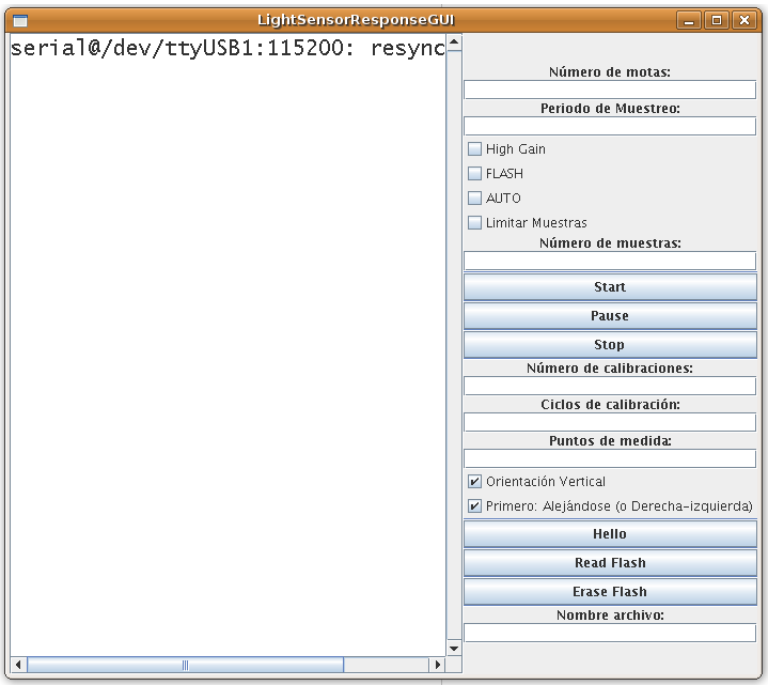


Figura 4.2: GUI para la calibración de los sensores de luz.

Como puede observarse en la imagen, se puede controlar el periodo de muestreo del sensor y el uso o no de la ganancia. Además, se pueden tomar datos de forma continua o limitando el número de muestras. Por otra parte, permite utilizar la memoria flash de la mota para almacenar los datos o simplemente enviar las muestras directamente al nodo base, que hace de interfaz entre el ordenador y los sensores. El uso de la memoria flash está especialmente pensado para utilizar varias motas simultáneamente y garantizar que no se pierden datos por colisión de mensajes al tratar de enviarlos al nodo base. Al tener los datos almacenados en la memoria flash se garantiza que éstos no se perderán en caso de que la mota se quede sin pilas o se apague por alguna razón. Es importante borrar la memoria antes de cada sesión de calibración para obtener los datos correctamente.

Otro detalle a comentar son las dos casillas seleccionables al final de la interfaz, que sirven para especificar la orientación de la mota con respecto a la forma en que se realiza la calibración. Esto es importante para poder adquirir los datos posteriormente en Matlab de forma automática. La orientación 1 implica que se toman muestras en perpendicular al sensor, alejándose en el primer ciclo y acercándose en el segundo. La orientación 2 es igual que la anterior pero en el primer ciclo el obstáculo se acerca. La orientación 3 implica que los datos se toman en paralelo al sensor, acercándose por su derecha en el primer ciclo, mientras que la orientación 4 supone datos tomados en paralelo y con aproximación desde su izquierda en el primer ciclo. En la Figura 4.3 se muestra un esquema de la disposición de las motas durante las sesiones de calibración, en ella los sensores están numerados según los criterios de orientación que se acaban de explicar.

Los datos recogidos se almacenan en un fichero individual para cada mota en un formato predefinido para su posterior procesado en Matlab.

4.2.2. Descripción de las Calibraciones

A la hora de realizar la calibración de los sensores de luz es importante tener en cuenta dos aspectos fundamentales: en primer lugar la respuesta de los sensores va a variar en función del tipo de iluminación (luz natural, luz artificial), y por otra parte será importante tener en cuenta la orientación del sensor respecto a la fuente de iluminación. Estas dos características determinarán el área de detección de un determinado sensor.

Para estudiar las dos características mencionadas, se realizaron diversas sesiones de calibración con cuatro sensores, uno por cada orientación descrita. En primer lugar se seleccionó un lugar en el que se pudiese medir el efecto tanto de la luz natural como de la luz artificial, y que fuese suficientemente amplio para tomar medidas a una distancia significativa. Por ello, se reservó una de las aulas y se despejaron los muebles para tener más libertad de movimientos. Se formó una cuadrícula de 7.2 m de largo y 3.6 m de ancho siguiendo las líneas marcadas por las baldosas del suelo, de manera que la separación entre puntos de medida era de 60 cm tanto a lo largo como a lo ancho, disponiendo por tanto de 13 puntos de medida a lo largo y 7 a lo ancho. En la Figura 4.3 se muestra un esquema de la disposición de los puntos de calibración y la posición de los sensores.

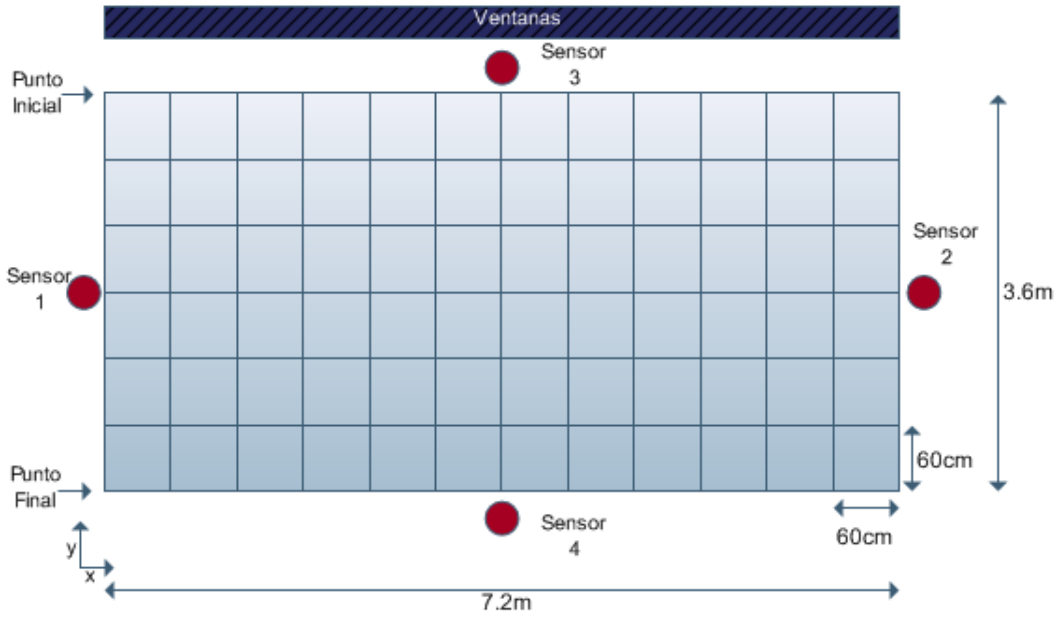


Figura 4.3: Esquema de la disposión de las motas durante las sesiones de calibración .

Para entender la forma en que se tomaron los datos, es importante definir los parámetros utilizados. Para ello se utiliza como referencia el *Sensor 1* tal como se muestra en la Figura 4.3.

Punto de medida: Cada uno de los puntos de la cuadrícula en los que se toman medidas.

Línea de referencia: Línea perpendicular al sensor a lo largo de la cual se va desplazando el objeto (persona en este caso). Siguiendo la nomenclatura del esquema anterior, cada valor $y = a$ constituye una línea de referencia.

Ciclo de calibración: Conjunto de todas las medidas tomadas en los puntos a lo largo de una misma línea de referencia, sin repetir puntos. Un ciclo de calibración estará compuesto por las medidas $(x_i, y), y = a, i = 1 : K$, siendo K el número de puntos a lo largo.

Calibración: Todas las medidas tomadas en una línea de referencia. Una calibración estará compuesta por uno o más ciclos de calibración.

Sesión de calibración: Todas las medidas tomadas bajo unas mismas condiciones de iluminación y con unos mismos parámetros. Cada sesión estará compuesta por varias calibraciones.

Para tomar las medidas se utilizó la función automática (AUTO) de la aplicación desarrollada, de manera que cada 5 seg se enviaba la orden para empezar a tomar las muestras. Para poder completar la sesión de calibración se especificarón las variables solicitadas en la interfaz: número

de calibraciones, número de ciclos de calibración, número de puntos de medida y número de muestras en cada punto. El único parámetro que se modificó de una sesión de calibración a otra fue el número de ciclos de calibración. En un principio se decidió utilizar cuatro ciclos por calibración, con el objetivo de tener un número mayor de datos y poder estudiar mejor la varianza de los mismos. Sin embargo, se observó que al tratar de calibrar la iluminación natural, ésta cambiaba demasiado durante una misma sesión, haciendo que la iluminación ambiente para la última calibración variase de forma muy apreciable con respecto a la primera. Por ello se decidió reducir el número de ciclos a dos. El resto de parámetros se mantuvo constante y con los siguientes valores:

- Número de muestras (N): 6
- Puntos de medida (numPtos): 13
- Número de ciclos de calibración (numCiclos): 2 (4 en las primeras sesiones)
- Número de calibraciones: 7

Para desplazarse por los puntos de medida, se tomó como posición inicial la indicada en la Figura 4.3, la esquina superior izquierda de la cuadrícula. A partir de este punto se realizaron dos ciclos en cada calibración, lo que supone desplazarse a lo largo de cada línea, primero yendo desde el sensor 1 al 2 para luego regresar y pasar a la siguiente línea, siendo el punto final el indicado en la mencionada figura.

Además de tomar medidas en los diferentes puntos, se tomaron al inicio y al final de cada sesión una serie de medidas de iluminación ambiente sin que hubiese ningún obstáculo en la cuadrícula definida. Calculando la iluminación media al inicio y al final de la sesión se puede ver la evolución a lo largo de la sesión, y en el caso de iluminación con luz natural donde la media varía considerablemente entre la primera y la última calibración, permite estimar la iluminación media para cada calibración individual y así normalizar los datos para obtener resultados más precisos. Este proceso se describe más adelante cuando se expongan los resultados obtenidos.

4.2.3. Resultados

El estudio de la respuesta de los sensores de luz está compuesto por varios análisis de los datos obtenidos. Antes de poder estudiar los resultados, fue necesaria una etapa de pre-procesado que se describe a continuación.

1. Los archivos .txt con los datos de cada mota se procesaron usando Matlab para obtener tanto los parámetros de la sesión como los datos. En esta etapa se eliminó la primera muestra de cada punto de medida, ya que el sensor proporcionaba valores del punto de medida anterior. También se calculó la media de las muestras en cada punto.

2. Una vez creada la estructura de datos, se calcularon los valores medios de las muestras para cada punto de referencia y en todos los ciclos de calibración. Así, si la sesión contaba con dos ciclos de calibración, se dispone de dos grupos de muestras en cada punto de medida, de forma que al eliminar la primera muestra de cada punto, se cuenta con un total de diez medidas en cada punto de calibración, de las cuales se calcula la media como valor de referencia para dicho punto.

3. Con las medias ya calculadas, se realizó una interpolación por tres en las dos dimensiones. Esto aumenta el margen de medidas y permite ver mejor los resultados.

4. Normalización de los datos. Este proceso es importante en el caso de iluminación con luz natural, ya que, como se ha mencionado, la iluminación ambiente varía de forma apreciable a lo largo de la sesión. Para realizar la normalización de los datos se consideraron dos opciones. La primera considerando una variación lineal de la iluminación media a lo largo de cada sesión y la segunda realizando un estudio de iluminación a lo largo de diferentes horas del día y ajustando la curva obtenida para luego evaluarla en los instantes temporales de interés. Como se verá a continuación, este estudio demuestra que la variación puede considerarse lineal en intervalos cortos de tiempo.

El resto de esta sección está organizada de la siguiente forma. En primer lugar se muestran los resultados obtenidos para iluminación natural y artificial con los datos interpolados pero sin normalizar, con el fin de justificar la necesidad de la mencionada normalización. A continuación se describe el proceso de normalización, mostrando los resultados tras aplicarlo a los mismos ejemplos. Finalmente, se realiza un estudio sobre el efecto del umbral de detección sobre el área de detección.

A lo largo de toda la sección se hará referencia a tres pruebas, cuyos detalles son los siguientes:

Prueba 1: Iluminación natural, 12.30 de la mañana.

Prueba 2: Iluminación natural, 18.30 de la tarde.

Prueba 3: Iluminación artificial, sin ninguna influencia de luz natural.

Resultados Sin Normalizar

El primer paso para analizar los sensores consiste en observar la respuesta de los mismos sin ningún tipo de normalización, con el fin de comprobar si realmente es necesaria.

Las Figuras 4.4, 4.5 y 4.6 muestran los resultados obtenidos en cada una de las motas. En cada gráfica se representa el valor medido por el sensor frente a la distancia a la que se encontraba el obstáculo.

La primera prueba (Figura 4.4) se realizó en torno a las 12.30 de la mañana. En teoría, las curvas deberían converger al valor de la iluminación ambiente en ausencia de obstáculos, sin embargo, se puede observar una disminución progresiva del nivel de iluminación ambiente a medida que se avanza en las calibraciones.

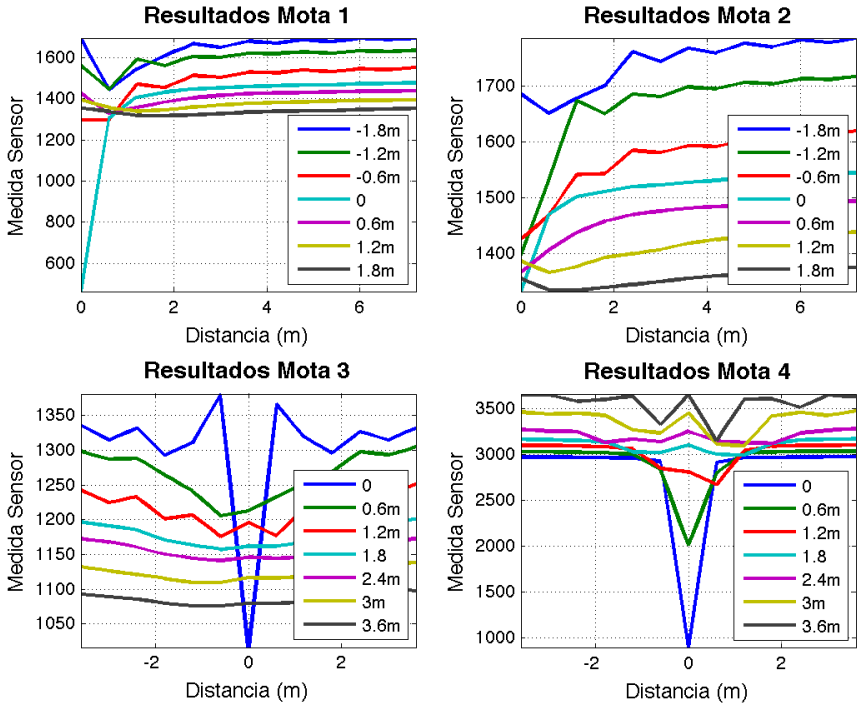


Figura 4.4: Resultados de la calibración con luz natural. 12.30 de la mañana.

En la Figura 4.5 se muestran los resultados de la sesión realizada en torno a las 18:30. Al igual que en los resultados anteriores, las curvas de calibración no convergen a un mismo valor, lo que indica de nuevo que existe una disminución progresiva del nivel de iluminación ambiente.

Finalmente, la Figura 4.6 proporciona los resultados para una sesión con iluminación artificial. En este caso, como era de esperar, el efecto de la variación de iluminación ha desaparecido y todas las calibraciones convergen a la misma iluminación ambiente media.

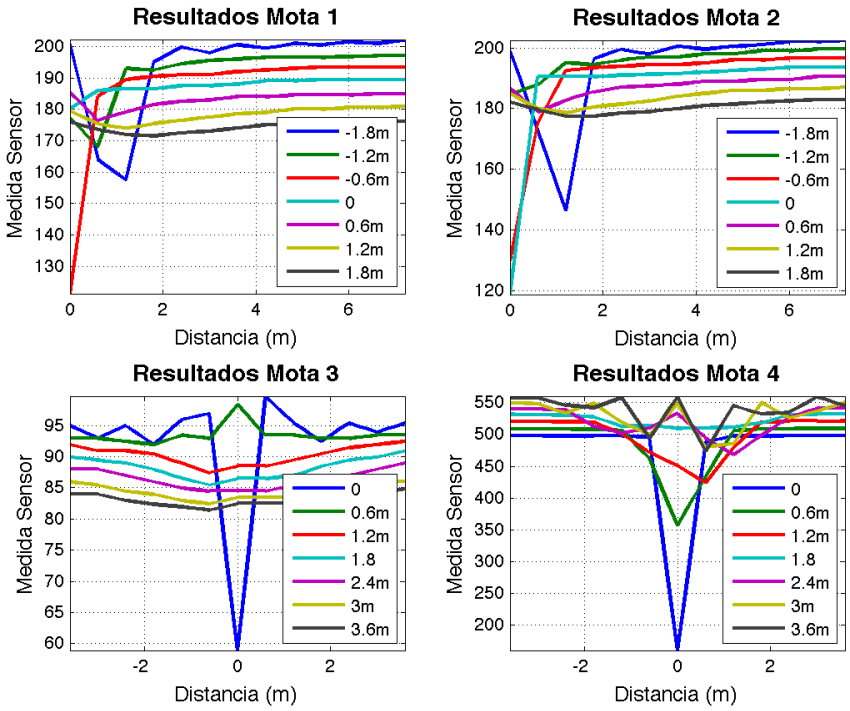


Figura 4.5: Resultados de la calibración con luz natural. 18.30 de la tarde.

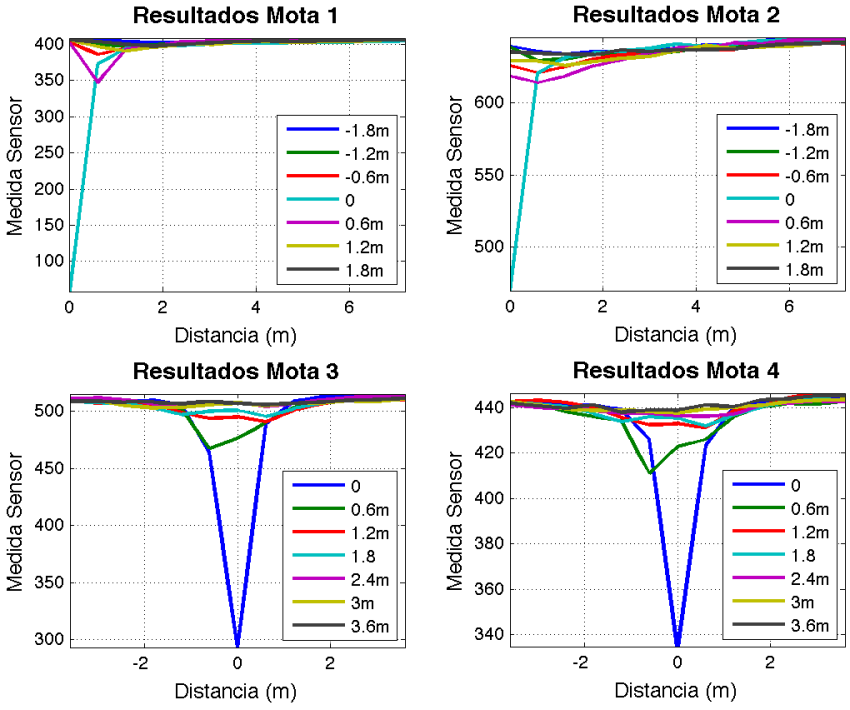


Figura 4.6: Resultados de calibración con luz artificial.

A la vista de los resultados queda justificada la necesidad de realizar un proceso de normalización que elimine la influencia de la variación de la iluminación en una misma sesión, de manera que todos los datos tengan como referencia el mismo valor de iluminación ambiente en ausencia de obstáculos.

4.2.4. Normalización

El problema de la normalización puede enunciarse de la siguiente forma: dados unos valores medios de iluminación ambiente al inicio y al final de una sesión, estimar el valor medio de la iluminación ambiente en ausencia de obstáculos para cada calibración perteneciente a esa misma sesión, y dividir los valores medidos en dichas calibraciones por el valor medio estimado.

Para poder estimar el valor medio, se dispone de los siguientes datos:

- amb_{init}, amb_{fin}: Valores medios de iluminación ambiente al inicio y al final de la sesión.
- h_{init}, h_{fin}: Hora en que se toman los datos de iluminación ambiente inicial y final.
- $h_{finCalib}$: Hora en que se termina la sesión de calibración.
- t_p: Tiempo de espera en cada punto de medida.
- $numCalib, numCiclos, numPuntos$: Número de calibraciones, ciclos por calibración y puntos de medida en cada ciclo.

La duración de cada calibración viene dada por la Ecuación 4.1.

$$t_{calib} = t_p \cdot numPuntos \cdot numCiclos \tag{4.1}$$

Para poder realizar la normalización es necesario conocer los instantes de tiempo de cada calibración individual, así como la ecuación que define la variación de la iluminación a lo largo de la sesión. Los instantes temporales se pueden calcular mediante la Ecuación 4.2.

$$t_i = h_{finCalib} - i * t_{calib} \qquad i = 1 : numCalib \tag{4.2}$$

A la hora de realizar un ajuste de los datos de iluminación ambiente para obtener la ecuación que define su comportamiento, se barajaron dos posibilidades: la primera considera una variación lineal durante el tiempo que dura la sesión de calibración, mientras que la segunda trata de

modelar la variación de la iluminación durante un periodo de tiempo más largo utilizando los datos de iluminación ambiente de diferentes sesiones a lo largo del día.

La Figura 4.7 muestra la evolución de la iluminación durante las pruebas cuyos resultados se presentaron anteriormente considerando dicha variación lineal. Como era de esperar, durante la sesión con iluminación artificial no existe ninguna variación, sin embargo, en las pruebas con iluminación natural se aprecia una disminución progresiva de la misma.

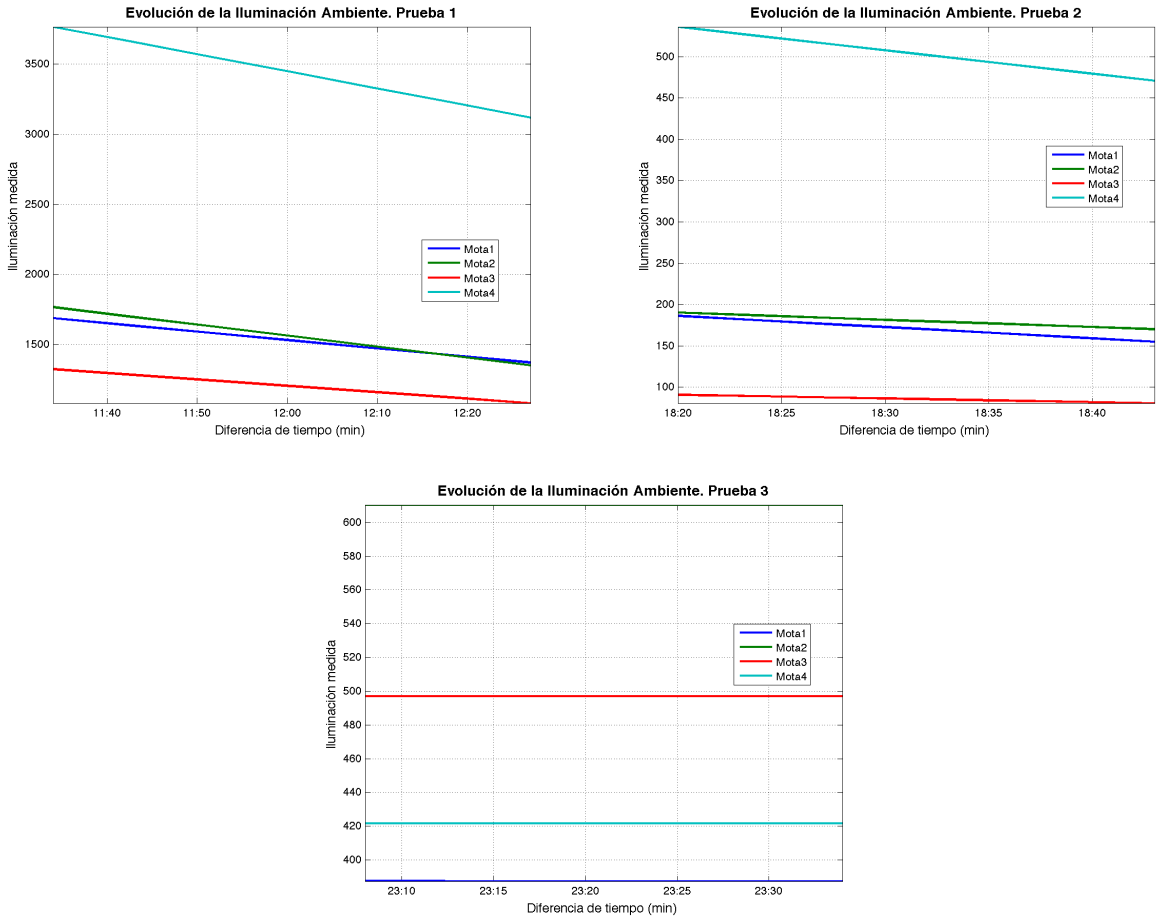


Figura 4.7: Evolución de la iluminación ambiente.

Si se consideran los resultados obtenidos en diferentes horas del día, se obtiene la curva de la Figura 4.8. De ella se puede concluir que para intervalos de tiempo cortos, la variación de la iluminación puede considerarse lineal. Teniendo en cuenta la duración de las sesiones de calibración (alrededor de 15 minutos), podemos asumir linealidad en la evolución de la iluminación ambiente.

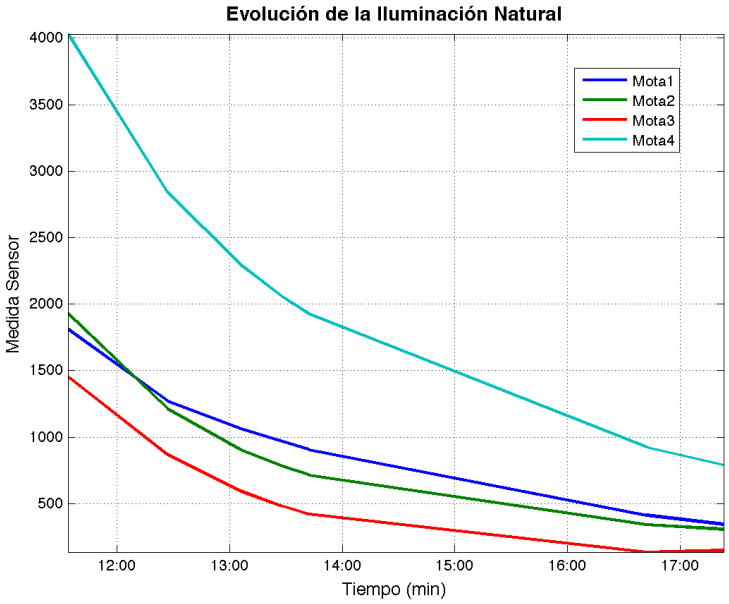


Figura 4.8: Evolución de la iluminación con luz natural a lo largo del día.

Es importante tener en cuenta que la única forma de obtener resultados muy precisos en cuanto a normalización sería tomar medidas de la iluminación ambiente en ausencia de obstáculos a la vez que se toman las medidas obstaculizando el sensor. Puesto que esto es imposible, en los resultados normalizados pueden aparecer efectos distintos a los esperados debidos por ejemplo, al efecto de las nubes en momentos puntuales.

Las Figuras 4.9 y 4.10 muestran los resultados tras la normalización para las dos pruebas con iluminación natural presentadas anteriormente.

Se puede observar el efecto de las ventanas de la sala sobre los resultados. Si la iluminación fuese uniforme, el mínimo de iluminación medida para cada calibración sería siempre el correspondiente al punto justo frente al sensor. Sin embargo se puede observar que el mínimo medido depende de la posición del obstáculo respecto al sensor y a las ventanas. Fijándose en la Mota 4, en la línea de calibración más cercana al sensor el mínimo sí se corresponde con el punto justo frente al mismo, pero al alejarse, se corresponde con el punto de máxima obstrucción de las ventanas, ya que justo frente al sensor no hay ventanas, de manera que no se obstruye la fuente de iluminación.

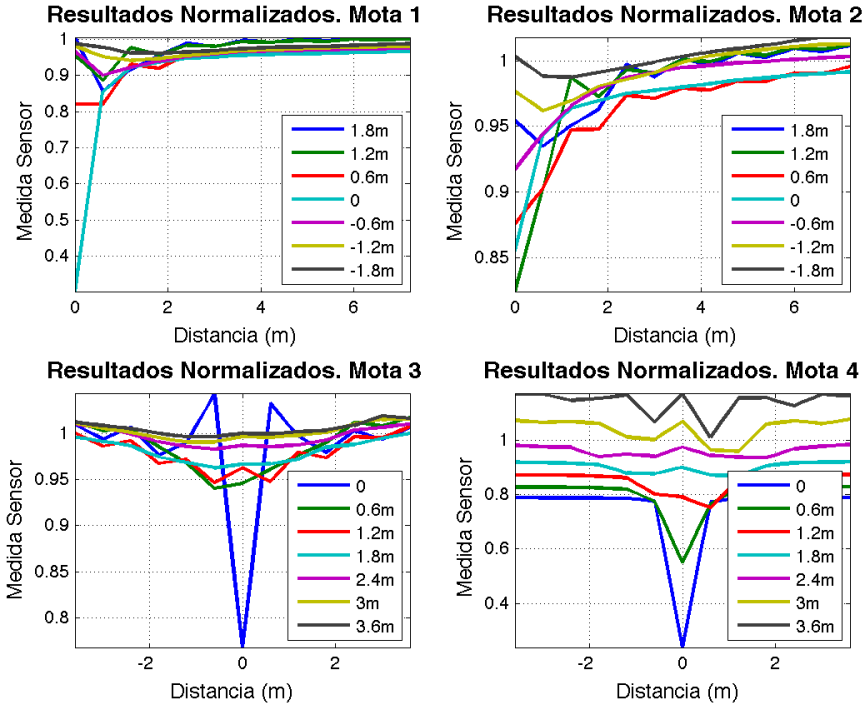


Figura 4.9: Resultados normalizados para la Prueba 1.

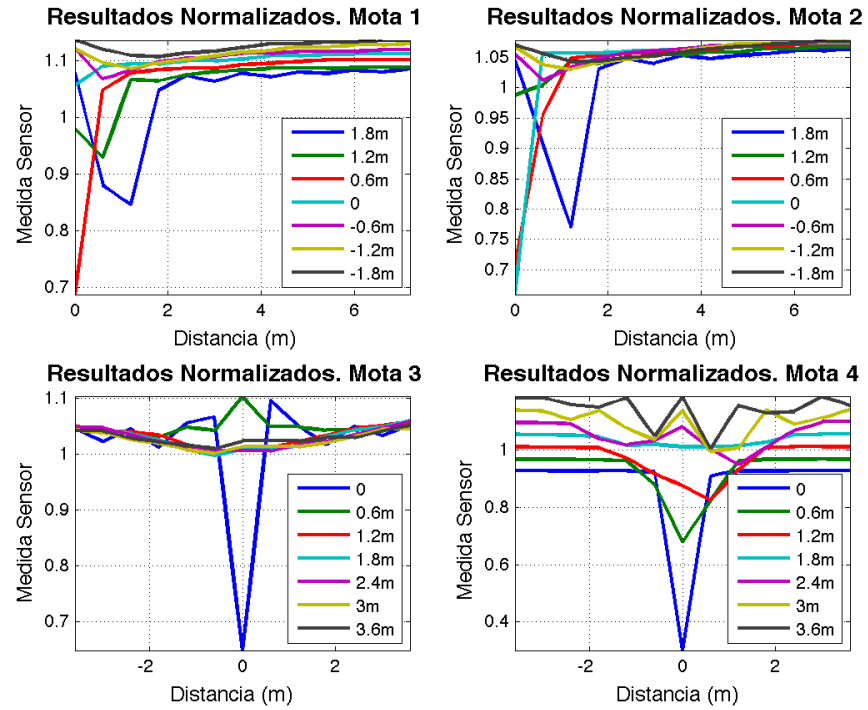
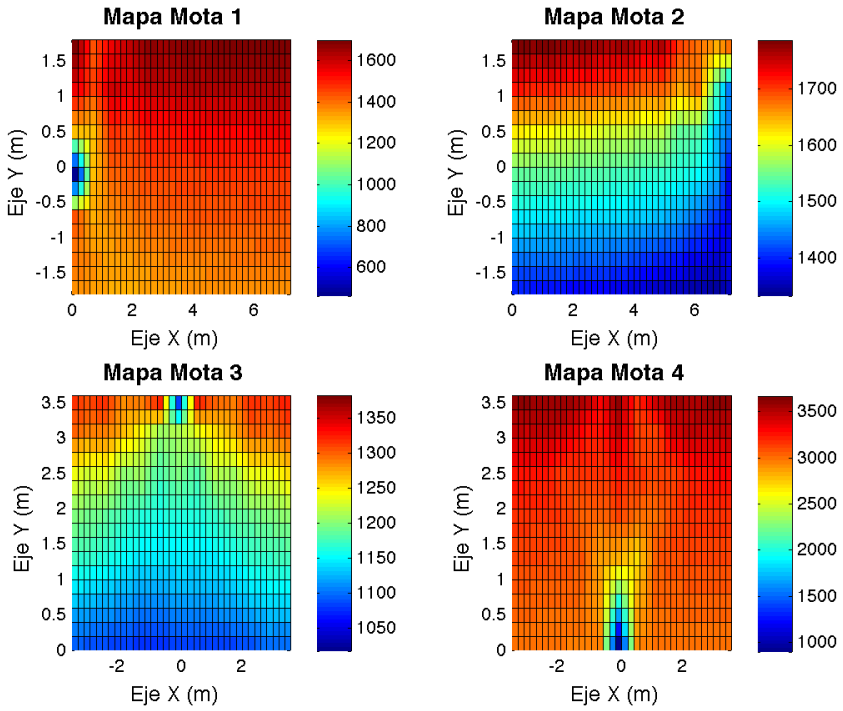


Figura 4.10: Resultados normalizados para la Prueba 2.

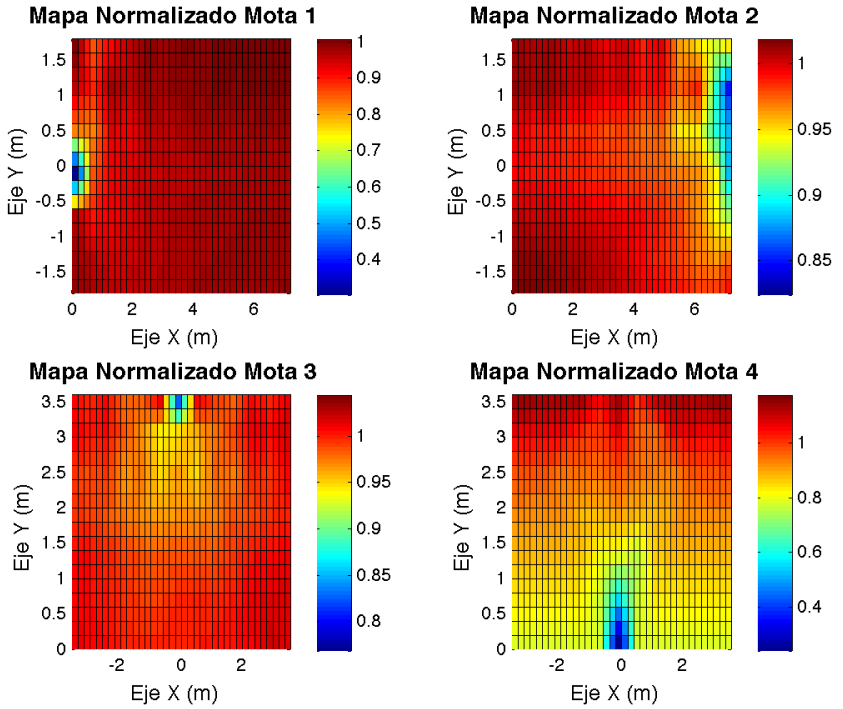
Para poder observar mejor las diferencias entre los resultados antes y después de la normalización, se presentan los mapas de iluminación en dos dimensiones para cada una de las pruebas utilizadas hasta ahora, antes y después de la normalización. Además, esta representación resulta muy útil para obtener una primera impresión del área de detección da cada sensor. En las Figuras 4.11, 4.12 y 4.13 se muestran dichos mapas, en los que cada mota se ha representado de acuerdo a su posición en el esquema mostrado en la Figura 4.3.

Si se observan los resultados mostrados para los dos primeros casos, en los que existía iluminación natural, el efecto de la variación de la iluminación resulta muy evidente. Por ejemplo, según los resultados de la Figura 4.11 a), el sensor 3, que se encontraba situado en la misma pared en la que estaban las ventanas, detecta mejor cuanto más lejos se encuentre la persona del sensor, lo cual no tiene sentido. Sin embargo, si se observa el resultado para este mismo sensor tras la normalización, lo que se obtiene es un área de detección razonable, con un mínimo de intensidad detectada justo frente al sensor y pegado a él, y con un aumento progresivo de la iluminación medida al alejarse del mismo.

En la última prueba (Figura 4.13) los resultados al normalizar no varían, ya que en este caso se disponía de iluminación artificial y por tanto la normalización no es necesaria.

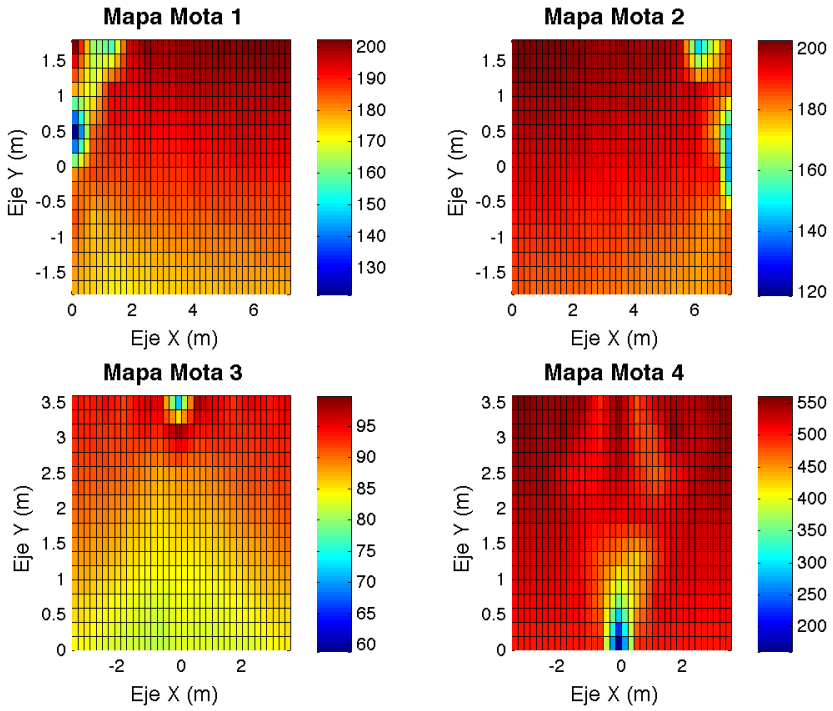


a) Sin normalizar.

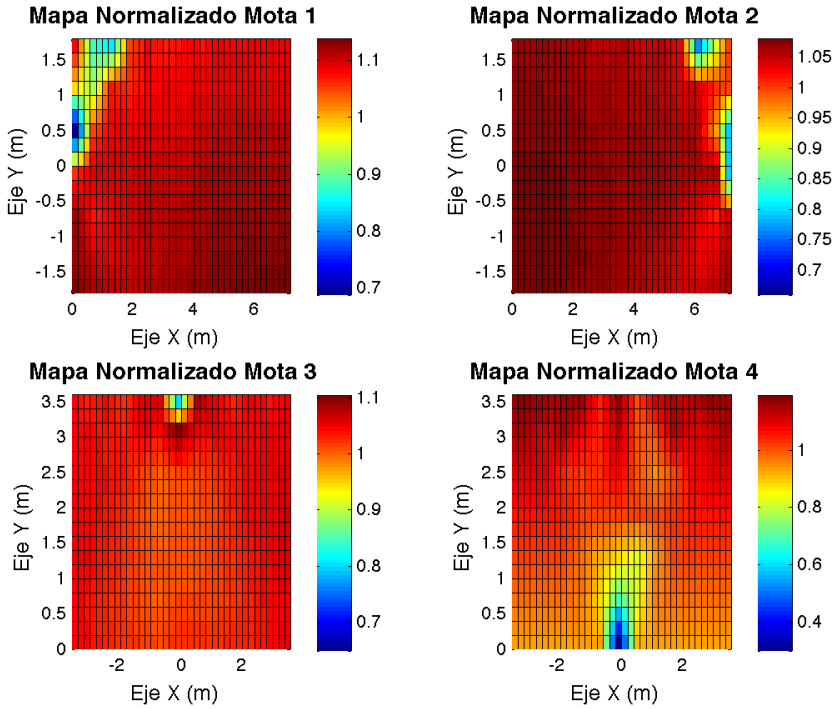


b) Normalizado.

Figura 4.11: Mapas de iluminación de la Prueba 1.

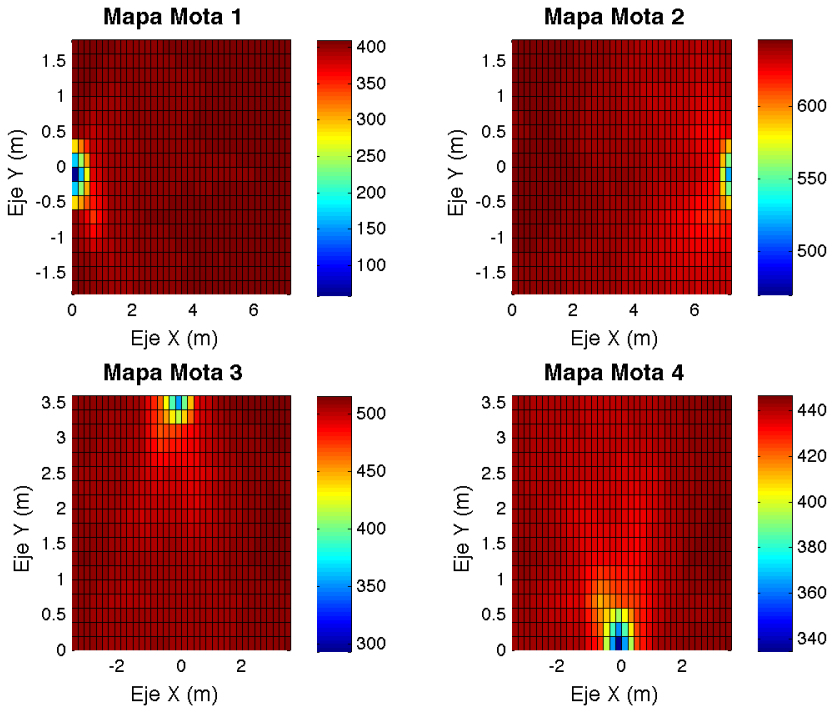


a) Sin normalizar.

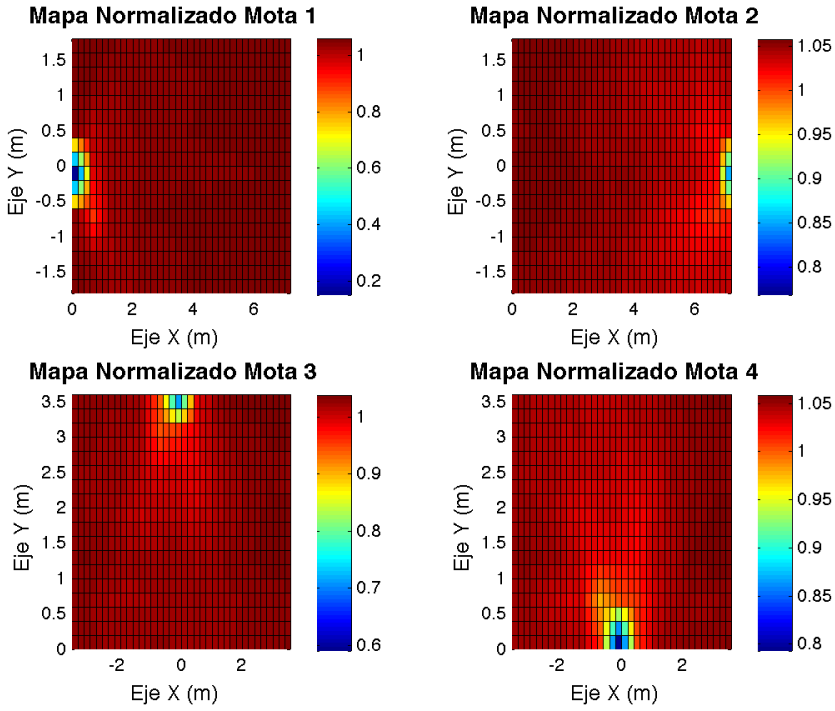


b) Normalizado.

Figura 4.12: Mapas de iluminación de la Prueba 2.



a) Sin normalizar.



b) Normalizado.

Figura 4.13: Mapas de iluminación de la Prueba 3.

4.2.5. Estudio del Umbral de Detección

El siguiente paso en la calibración de los sensores lumínicos consiste en estudiar qué umbral de detección debería fijarse. Este umbral será un porcentaje de variación con respecto a la iluminación media. Un porcentaje de variación excesivamente alto dará lugar a una región de detección muy pequeña y por tanto a una alta probabilidad de falsos negativos. Por el contrario, un umbral muy bajo resultará en una región de detección más amplia pero la probabilidad de falsos positivos aumentará.

En las Figuras 4.14, 4.15 y 4.16 se muestra la superposición de los mapas de detección de los cuatro sensores utilizando distintos umbrales: 10, 5, 2 y 1 % con respecto a la iluminación media.

Un mapa de detección refleja las zonas en las que un determinado sensor es capaz de detectar de acuerdo con un umbral determinado. Combinando los mapas de cada sensor se obtienen los resultados mostrados.

Para obtener el mapa de detección de una determinada prueba, primero se calculó el mapa individual de cada sensor. Para ello se obtuvo una matriz de resultados binaria para cada una de las motas a partir de la matriz de datos de las medidas originales, aplicando el siguiente algoritmo:

```
%N = Número de puntos de medida en el eje Y.
%M = Número de puntos de medida en el eje X.
matrizBinaria = zeros(N,M)
for i = 1:N{
    for j = 1:M{
            if matrizOriginal(i,j) < alpha
                    matrizBinaria(i,j) = 1;
        }
}
```

Con las cuatro matrices binarias, se puede obtener el mapa de detección combinado, representado por una matriz con cinco posibles valores que van del 0 al 4 y que reflejan el número de sensores que detectaron la presencia del obstáculo en cada punto concreto. Así, el mapa de detección final es la suma de las matrices binarias.

Como era de esperar, el área de cobertura varía en función del umbral de detección y de la iluminación ambiente. Así, los mejores resultados se obtienen para luz natural en las horas con más sol.

Los resultados de la Prueba 1 (Figura 4.14) muestran que se puede garantizar un 100 % de cobertura con un umbral razonable utilizando únicamente cuatro sensores. A medida que la iluminación baja, como ocurre en la Prueba 2, la cobertura para los mismos umbrales disminuye de forma muy apreciable. Si en lugar de iluminación natural ésta es artificial, resulta imposible cubrir el 100 % del área sólo con cuatro sensores, incluso con umbrales de detección muy pequeños.

Estos resultados demuestran que para garantizar una buena cobertura en cualquier circunstancia de iluminación sería necesario utilizar una densidad de sensores mucho mayor.

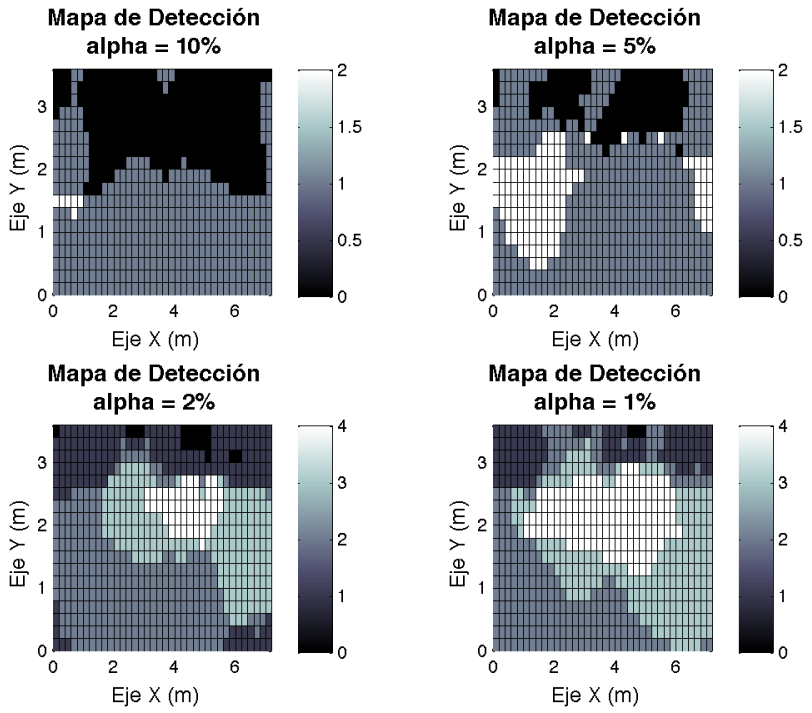


Figura 4.14: Mapa de detección para la Prueba 1 con diferentes umbrales.

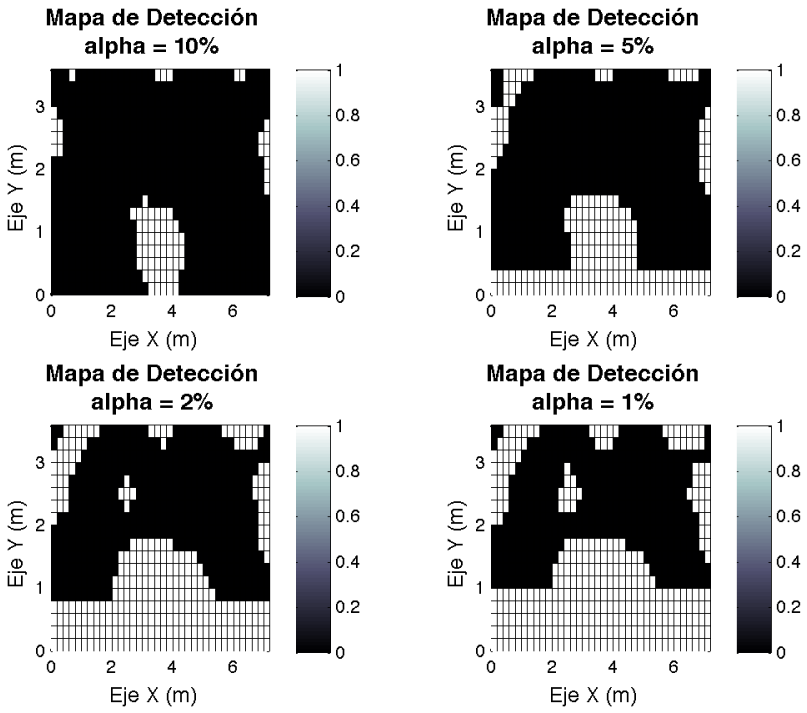


Figura 4.15: Mapa de detección para la Prueba 2 con diferentes umbrales.

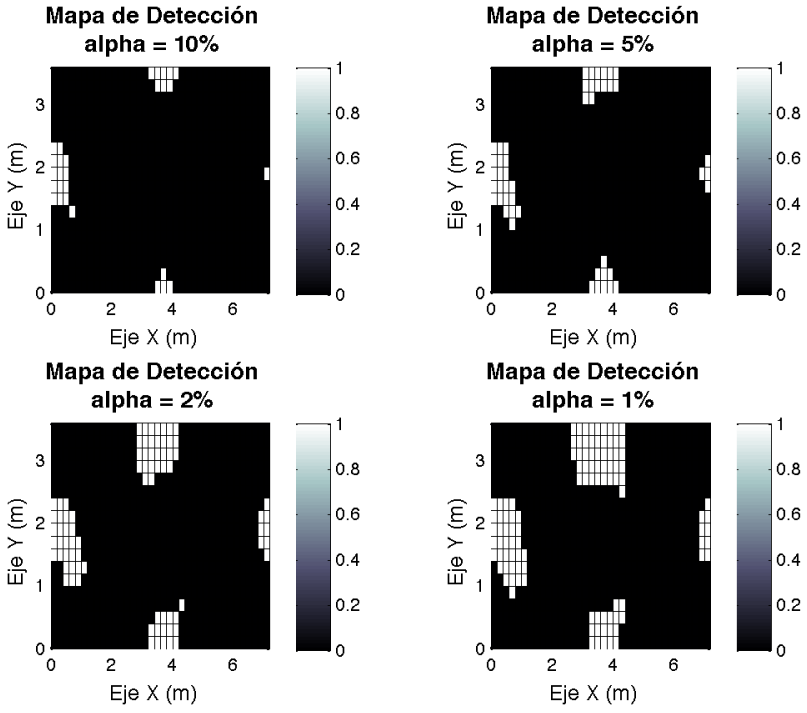


Figura 4.16: Mapa de detección para la Prueba 3 con diferentes umbrales.

Para decidir el umbral óptimo, se analiza la varianza de la iluminación cuando no existe ninguna obstrucción. Para ello, se utilizan los datos de iluminación ambiente obtenidos antes y después de cada una de las pruebas realizadas. En la Tabla 4.2 se muestra la varianza, expresada como porcentaje de la iluminación media, para cada una de las pruebas y cada uno de los sensores. Dicha varianza es la media de las obtenidas a partir de los datos de iluminación ambiente al inicio y al final de cada sesión.

Tabla 4.2: Varianza de la iluminación ambiente.

	Sensor 1	**Sensor 2**	**Sensor 3**	**Sensor 4**
Prueba 1	0.1352	0.1973	0.1590	0.3384
Prueba 2	0	0.0407	0	0.0464
Prueba 3	0.8284	0.9850	0.7342	1.0345
Media	0.3212	0.4077	0.2977	0.4731

Como puede observarse en la tabla anterior, la varianza media es menor al 1 % de la iluminación ambiente media. Si se tiene en cuenta además que el periodo de muestreo de los sensores de luz será generalmente inferior a 500 ms, y que por tanto la probabilidad de que la iluminación varíe en una cantidad mayor entre dos muestras consecutivas es muy pequeña, se puede concluir que es posible utilizar un umbral de un 1 % maximizando la probabilidad de detección y manteniendo la probabilidad de falsos positivos baja.

4.3. Sensores Acústicos

Para realizar un análisis contextual sencillo dentro de la aplicación, se utilizaron los micrófonos y el códec de audio incluido en la placa IMB400. La idea básica consiste en grabar datos continuamente calculando la energía de los mismos y detectando picos de energía atípicos que se puedan asociar con una posible caída del usuario.

A continuación se proporciona una descripción más detallada de estos sensores y se presentan los resultados de calibración.

4.3.1. Descripción General

La placa multimedia cuenta con un códec de audio WM8940, además de micrófono y altavoz integrados, permitiendo la captura y reproducción de audio sin necesidad de componentes

externos. Las características fundamentales del codec son las siguientes:

- Frecuencia de muestreo configurable: 8, 11.025, 16, 22.05, 24, 32, 44.1 y 48 kHz.
- SNR >94dB, THD <-82dB.
- Filtro paso alto configurable.
- 4 notch-filters (uno de ellos configurable como filtro paso bajo).
- Control automático de ganancia en la entrada al ADC y control automático de nivel para mantener el volumen de grabación constante.
- Ajuste de volumen de salida.

De las múltiples opciones de configuración disponibles para el códec, las que resultan realmente relevantes para la aplicación son la frecuencia de muestreo y los filtros.

El objetivo final consiste en establecer un umbral en función de los parámetros utilizados que permita discriminar los eventos anómalos. Para ello será necesario en primer lugar filtrar la señal con el fin de analizar únicamente las bandas de frecuencia de interés. Una vez se ha filtrado la señal, será necesario utilizar algún tipo de medida de análisis que permita establecer un umbral de detección, en este caso será la denominada *Short Time Energy, (STE)*. A continuación se muestran los resultados obtenidos y se determinan los parámetros que será necesario fijar en la aplicación.

4.3.2. Calibración

El objetivo de la calibración de los micrófonos es determinar la respuesta de los mismos ante diferentes sonidos que se produden a lo largo del día en una casa en la que habita una sóla persona, como pueda ser una llamada de teléfono, un objeto que se cae al suelo, pasos o la apertura de una puerta, con el fin de poder distinguir una posible caída del resto de sonidos.

Hacer un análisis completo y preciso de todos los posibles sonidos que se pueden producir en una casa es una tarea compleja y que escapa a los límites del presente proyecto. El problema se ha simplificado con el fin de mostrar cuál sería el proceso y simplemente se han grabado una serie de sonidos representativos a diferentes frecuencias de muestreo y se ha estudiado su espectro y su energía. Utilizando la información del espectro se podrán configurar los filtros necesarios, mientras que con la información de energía se podrá fijar un umbral aproximado. Estas dos ideas se describen en más detalle a lo largo de esta sección.

Para las pruebas con los sensores acústicos se desarrolló una aplicación que permite seleccionar la frecuencia de muestreo, utilizar los filtros seleccionando la frecuencia de corte de los mismos y configurar el enventanado para el cálculo de la energía. En la Figura 4.17 se muestra la interfaz gráfica de dicha aplicación.

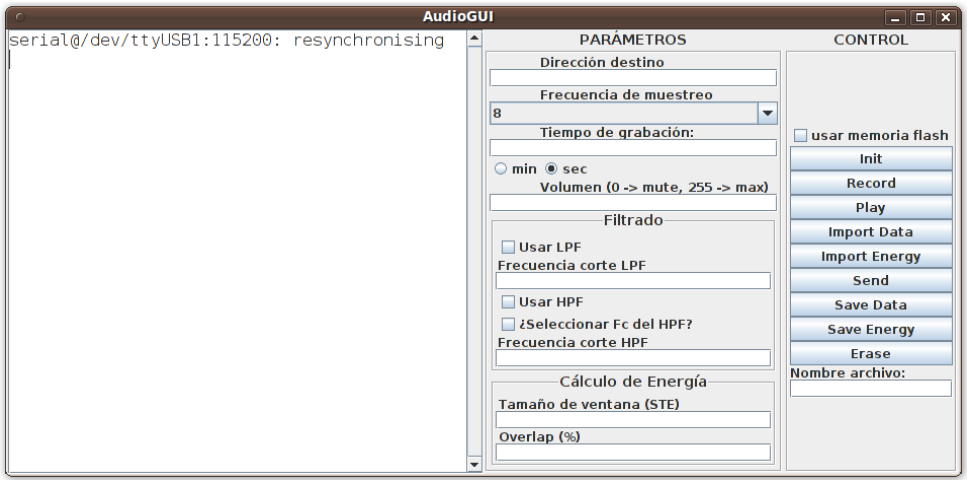


Figura 4.17: GUI de la aplicación para analizar sonido.

De acuerdo con las hojas de catálogo del códec de audio, el filtro paso alto puede funcionar en dos modos: *audio*, en el que el filtro es de primer orden con frecuencia de corte entorno a los 3.7 Hz, y *application*, en el que el filtro es de segundo orden con una frecuencia de corte seleccionable de entre una serie de valores predefinidos. El filtro paso bajo es un filtro de primer orden cuya frecuencia de corte no está predefinida.

Para estudiar los sensores acústicos se grabaron con éstos una serie de sonidos con el objetivo de observar la señal recibida y determinar su espectro. Con ello se podrán seleccionar adecuadamente las frecuencias de corte de los filtros, así como un umbral de detección basado en la energía. Como señales representativas se han seleccionado las siguientes:

- Ruido.
- Pasos. Se grabaron pasos acercándose y alejándose del sensor.
- Llamada de teléfono. Se graba tanto el sonido del móvil como la converación.
- Llamada de teléfono. Sólo el tono de llamada.
- Caída de un objeto pequeño (un bolígrafo) a 2 m y a 4 m del sensor.
- Golpe simulando una caída a 2 m y a 4 m del sensor.

Short Time Energy (STE)

Para poder establecer un umbral de detección se ha utilizado el valor de la energía a corto plazo, STE. Este parámetro se usa frecuentemente en el análisis de señales de audio para distinguir fragmentos de voz y de silencio, ya que la energía de estos últimos es mucho menor. La formulación es la siguiente:

Siendo d la señal de audio, y N el tamaño de ventana, el valor de la energía será:

$$e_n = \frac{1}{N} \sum_{k=1}^{N} d_k^2 \tag{4.3}$$

Para todas las pruebas realizadas, el tamaño de ventana seleccionado es de 50 ms, ya que es suficientemente pequeño para detectar cambios bruscos rápido y suficientemente grande para ser representativo.

Resultados

Una vez definida la energía a corto plazo, se muestran los resultados de las diferentes pruebas realizadas. Para estudiar las señales se muestran la señal grabada, la energía a corto plazo calculada en la mota y la estimación del espectro.

La señal grabada es una secuencia de valores de intensidad cuyo rango viene limitado por el códec de audio, que almacena cada muestra como un entero con signo de 16 bits, por lo que el rango de valores es de $[-32{,}768, 32{,}767]$. En cuanto a la estimación del espectro, se ha utilizado el método Welch [22] con 1024 puntos de FFT, 256 ventanas y un 50 % de solapamiento.

En la primera prueba se grabó el sonido ambiente durante 5 segundos a dos frecuencias: 8 y 22 kHz. No se probaron frecuencias más altas puesto que el consumo de energía es mayor cuanto más alta sea la frecuencia de muestreo, por lo que resulta conveniente utilizar la frecuencia más baja posible, además, las frecuencias de muestreo altas se suelen emplear para audio de alta calidad, lo cual no es necesario en este caso. En la Figura 4.18 se muestran los resultados de las señales grabadas junto con su energía y su espectro.

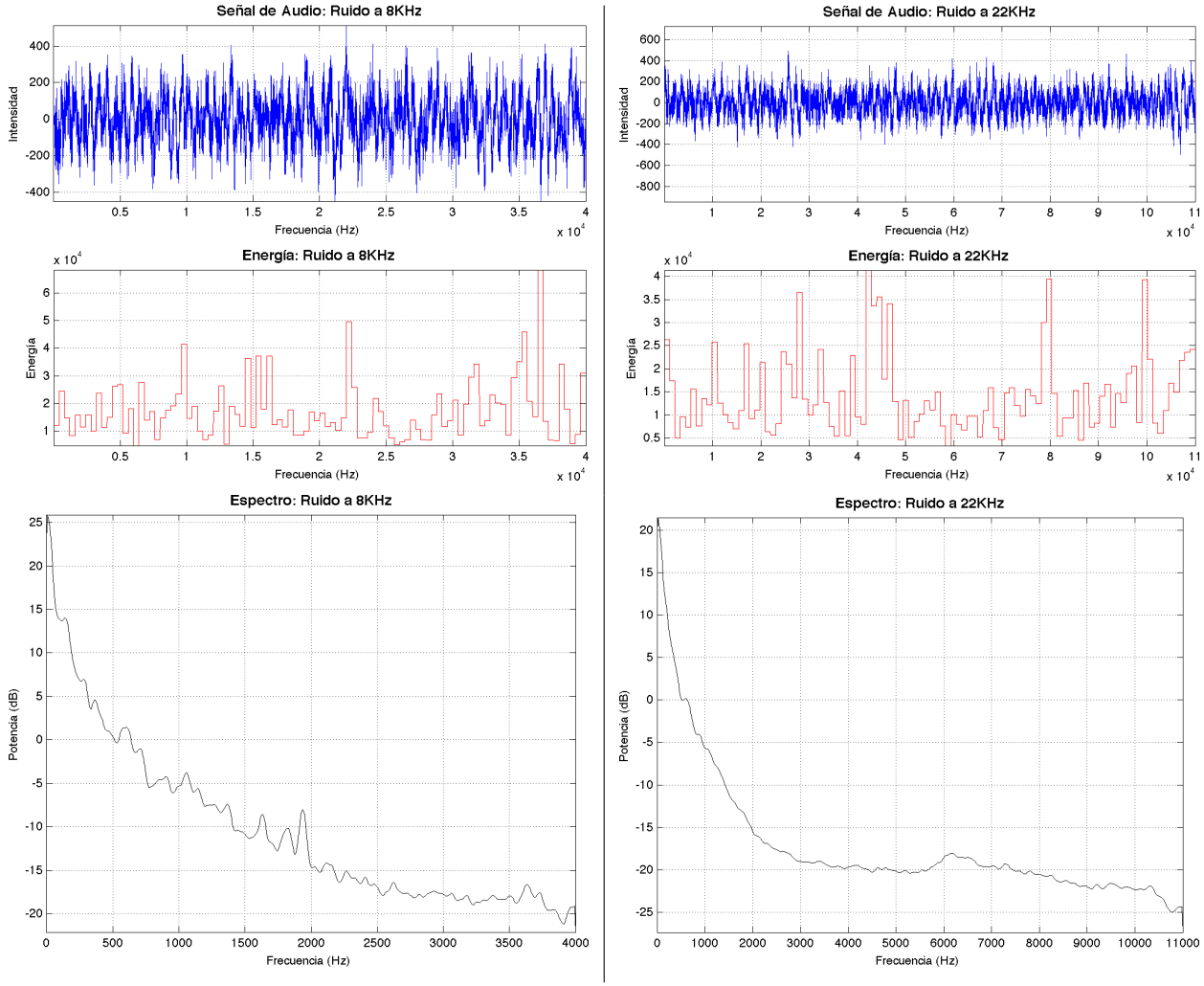


Figura 4.18: Señal, energía y espectro del ruido a 8KHz y a 22KHz

Como era de esperar, el ruido tiene una intensidad muy baja y su espectro está en bajas frecuencias, por lo que se podría mitigar su efecto utilizando un filtro paso alto que elimine las frecuencias inferiores a 10 Hz.

La segunda prueba consiste en grabar pasos de una persona que se acerca y se aleja del sensor. En este caso los pasos se grabaron sobre una superficie sin alfombras y con tacones, por lo que sólo es una muestra de las múltiples combinaciones que podrían darse. Para obtener datos más representativos sería necesario grabar la respuesta del sensor ante pasos en diferentes superficies, con diferente calzado y con personas diferentes.

Al igual que en el caso anterior, se realizó la prueba con 8KHz y 22 kHz como frecuencias de muestreo. La Figura 4.19 muestra los resultados obtenidos. Se puede observar que existe cierta periodicidad en la señal, la intensidad es relativamente baja en las pisadas y la mayor parte de la potencia se concentra en frecuencias inferiores a 1 kHz.

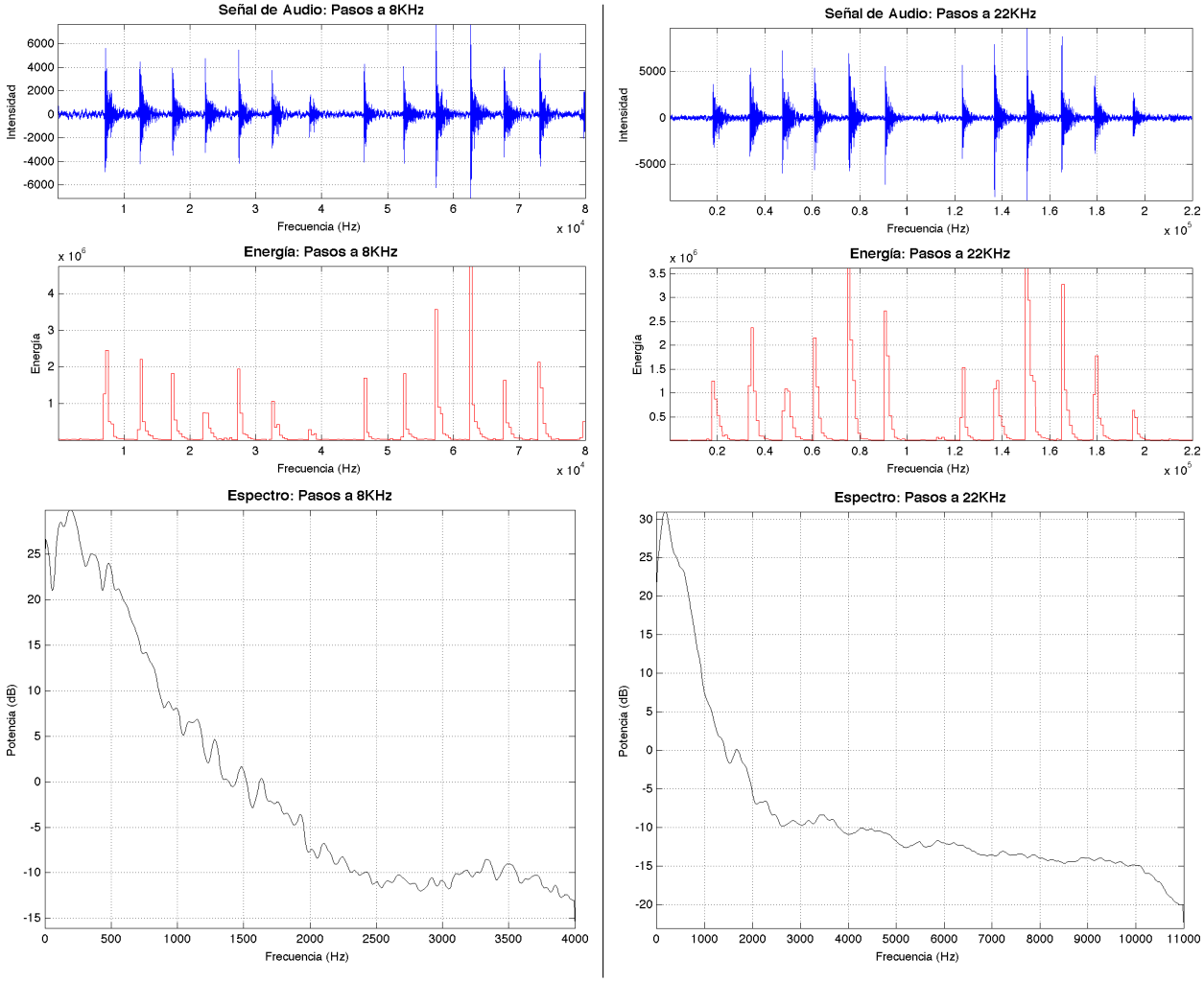


Figura 4.19: Señal, energía y espectro de pasos a 8 kHz y a 22 kHz

Una llamada de teléfono es algo de la vida cotidiana que suele producirse todos los días. Por ello se realizaron dos pruebas basadas en esta situación. En la primera (Figura 4.20) se grabó el sonido de una llamada a un móvil con un tono de llamada clásico. En la segunda (Figura 4.21) simplemente se grabaron, además del tono, unos segundos de conversación, de esta forma se puede observar el efecto de la voz sobre la señal, especialmente sobre su espectro.

En este caso sólo se utilizan 8 kHz como frecuencia de muestreo, ya que la diferencia para la aplicación de interés con una frecuencia de 22 kHz no justifica el aumento del consumo de energía.

Se puede observar que el tono de llamada de teléfono se encuentra en torno a los 1500 Hz, pero queda enmascarado cuando se añade voz a la señal, de ello se puede deducir que la potencia de las componentes espectrales del tono de teléfono es mucho menor que la de la voz, ya que en caso contrario la señal del teléfono no habría quedado enmascarada.

Cabe recordar que la voz se encuentra típicamente entre los 50 - 200 Hz para el hombre y 150 - 350 Hz para la mujer, de ahí que la mayor parte del espectro de la señal se encuentre en bajas frecuencias.

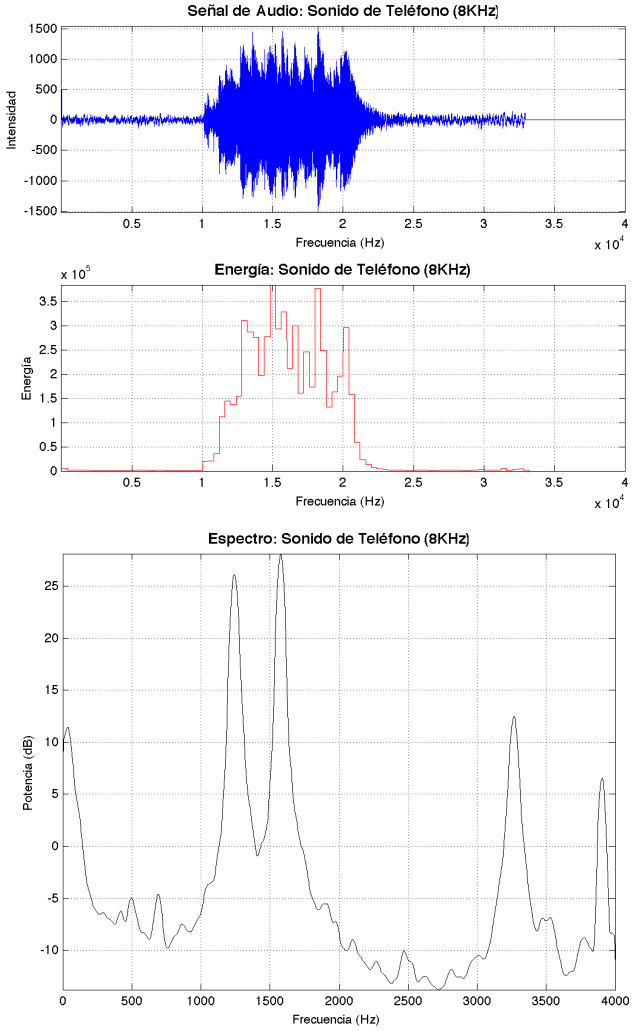


Figura 4.20: Señal, energía y espectro del tono de un teléfono.

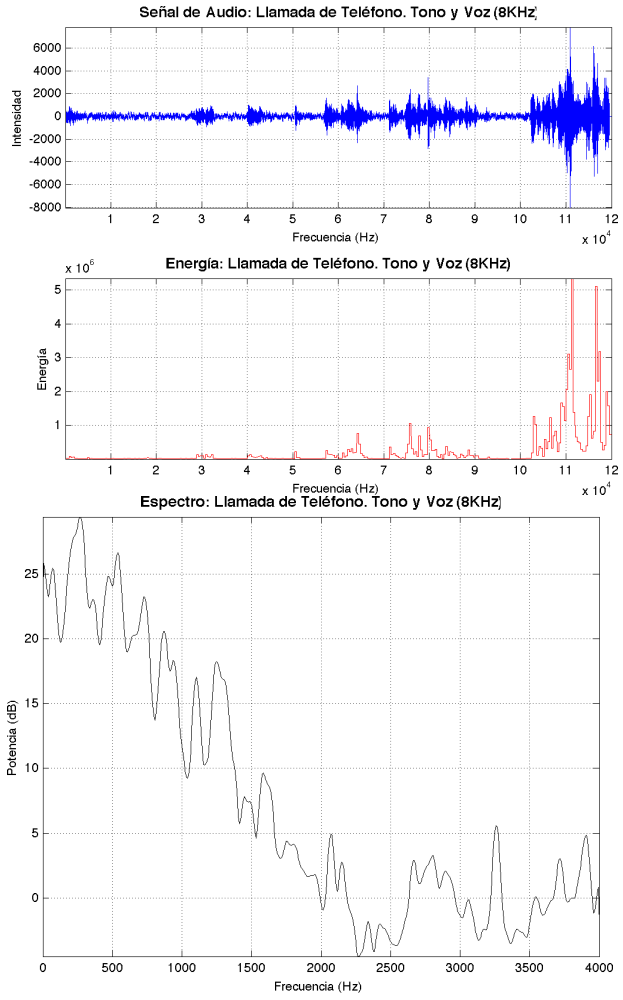


Figura 4.21: Señal, energía y espectro de una llamada de teléfono.

Otra situación típica en la vida cotidiana es la caída al suelo de algún objeto. De nuevo, sería necesario realizar múltiples pruebas para caracterizarla correctamente: diferentes superficies y diferentes objetos. Aquí simplemente se dejó caer un bolígrafo al suelo, a 2 m y a 4 m de donde estaba situado el sensor acústico.

Los resultados se muestran en la Figura 4.22, donde se puede observar que de nuevo predominan las bajas frecuencias y que la distancia es relevante en la intensidad.

Por último, se simuló una caída como un golpe fuerte de un objeto pesado que se cae al suelo. Se realizó la prueba a 2 m y a 4 m del sensor, y se obtuvieron los resultados de la Figura 4.23, en los que se puede observar una situación semejante a la anterior, con valores de intensidad y de energía mayores.

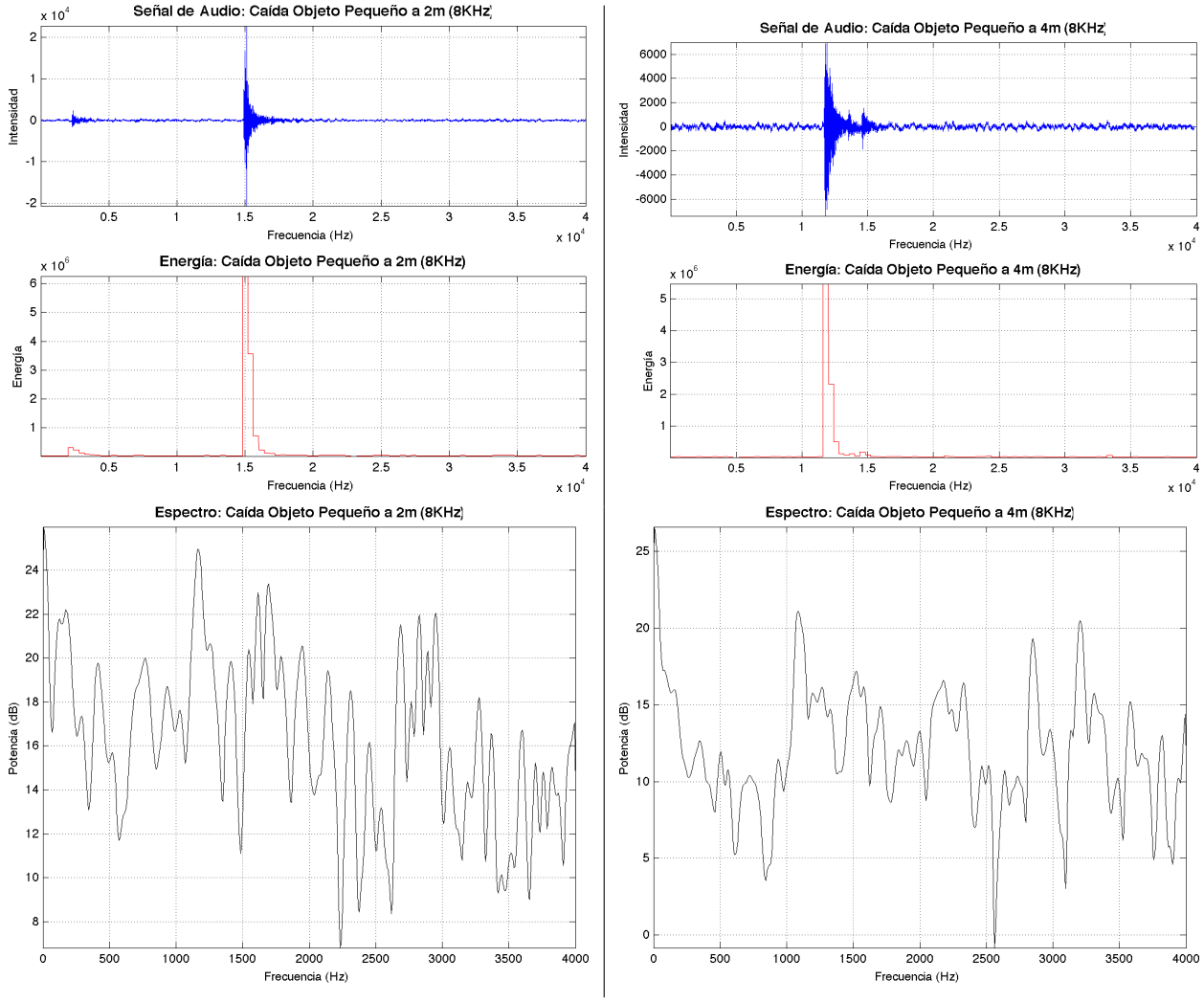


Figura 4.22: Señal, energía y espectro de un objeto pequeño que se cae al suelo a 2 m y a 4 m

Evidentemente una caída real se puede producir de muchas formas: una caída seca al suelo, un tropiezo con algún objeto, una caída que incluya que se mueva por ejemplo una silla u otro objeto al caer, etc, además de poder producirse en distintas superficies: parquet, moqueta, alfombras o baldosas generalmente. Por ello, caracterizar adecuadamente una caída supondría realizar un estudio detallado de todas estas posibilidades. Por otra parte, es muy difícil simular una caída de forma realista. Aquí simplemente se realizó una simulación que equivaldría a una caída sobre baldosas, sin objetos alrededor.

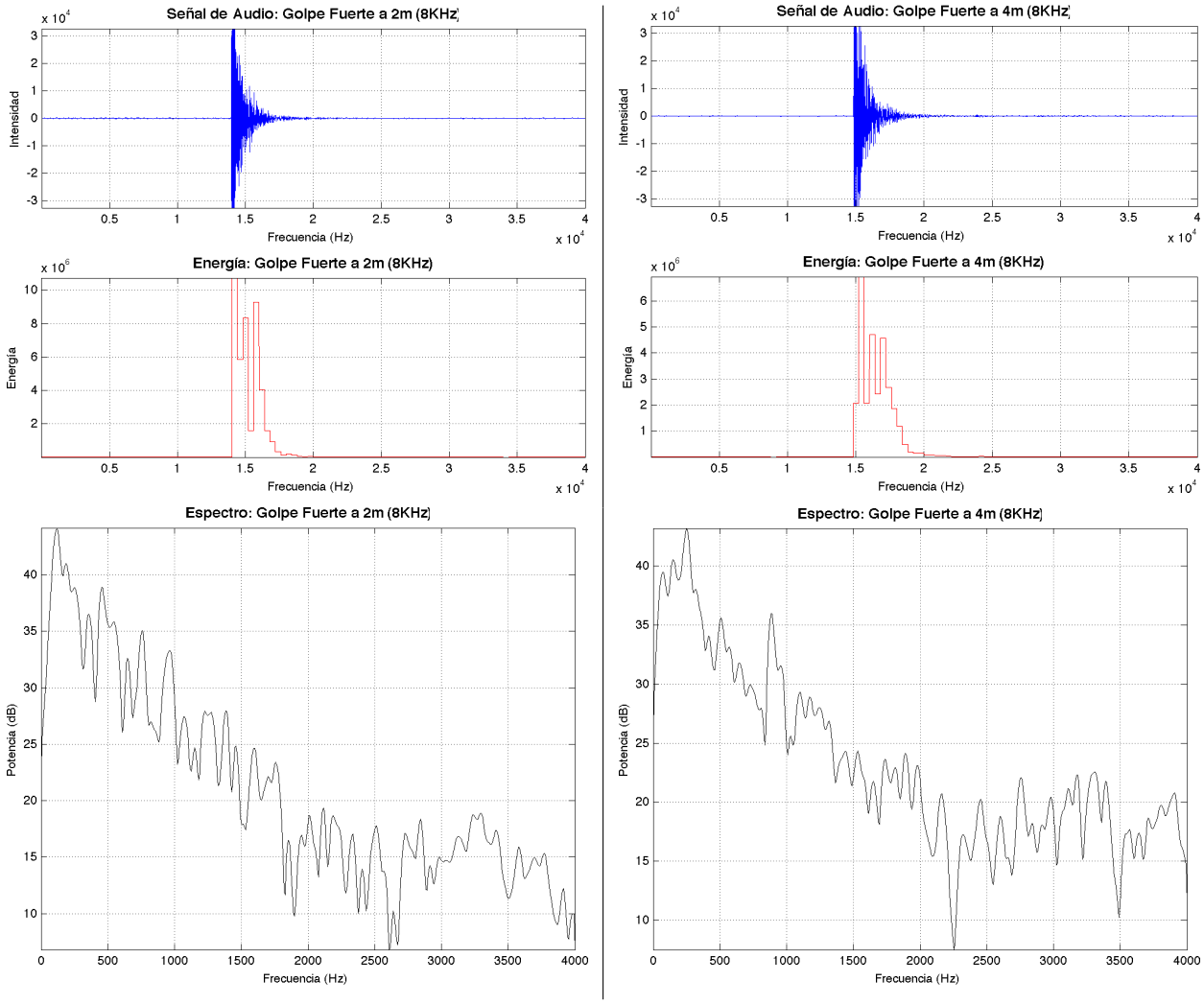


Figura 4.23: Señal, energía y espectro de un golpe seco a 2 m y a 4 m

Los resultados vistos hasta ahora muestran señales muy diferentes en su envolvente, pero con espectros similares. Para comparar los resultados de forma cuantitativa, se muestran en la Tabla 4.3 los valores máximos de intensidad, energía y potencia para todas las señales grabadas, junto con la frecuencia correspondiente a la máxima potencia (F_{potMax}).

Se puede observar que el valor de intensidad registrado por el sensor cuando se produce un golpe fuerte (simulando la caída) es el máximo posible, por lo que se puede deducir que la capacidad de discernir entre sonidos atendiendo a la intensidad va a quedar limitada por el rango del sensor acústico.

Por otra parte, los valores de energía son mucho mayores en el golpe a 2 m del sensor que en cualquier otra prueba, por lo que parece posible fijar un umbral de energía que permita discriminar un evento anómalo como una caída.

Finalmente, del análisis espectral se deduce que la mayoría de las situaciones propuestas tienen un espectro de potencia comprendido en bajas frecuencias. Para establecer mejor una comparación se han representado en la Figura 4.24 los espectros estimados en las pruebas anteriores con frecuencia de muestreo de 8 kHz. Además, para mayor simplicidad se eliminan las dos pruebas realizadas a 4 m del sensor.

Tabla 4.3: Valores máximos de las señales de prueba.

Sonido	F_s **(KHz)**	**Intensidad**	**Energía**	**Potencia (dB)**	F_{potMax} **(Hz)**
Ruido	8	513	68.486	25,89	7,81
Ruido	22	945	41.319	21,46	21,48
Pasos	8	7.678	4.751.011	29,87	195,31
Pasos	22	9.685	3.620.804	30,96	171,88
Tlf y Voz	8	8.041	5.339.856	29.42	265,65
Tono Tlf	8	1.545	466.077	28,11	1.578,1
Caída Objeto 2m	8	22.813	6.262.723	25,98	7,81
Caída Objeto 4m	8	7.395	5.476.912	26,59	7,81
Golpe 2m	8	32.768	10.732.974	44,16	117,19
Golpe 4m	8	32.768	6.934.175	43,19	250

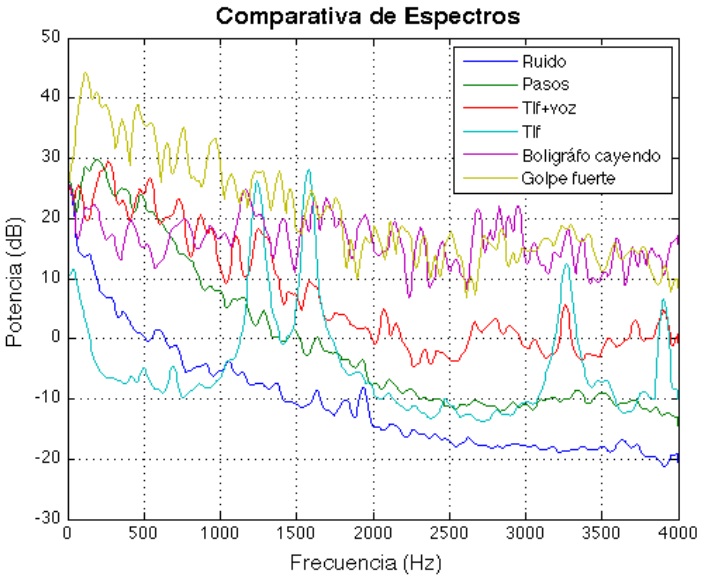


Figura 4.24: Comparativa de espectros.

Se puede observar que pese a tener espectros semejantes, la potencia de las componentes frecuenciales del golpe fuerte es mayor que el resto. De esta figura se puede deducir que las altas frecuencias resultan irrelevantes para los sonidos de interés, por lo que sería conveniente utilizar un filtro paso bajo que las elimine. Por otra parte, se podría eliminar el ruido de baja frecuencia mediante un filtro paso alto adecuado.

A partir de las pruebas realizadas se puede concluir que un sistema de análisis contextual basado únicamente en un umbral de energía sobre una señal filtrada apropiadamente puede dar lugar a una probabilidad de falsos positivos muy alta. Para reducir esta probabilidad sería necesario realizar un estudio mucho más detallado de los posibles sonidos que permita implementar un clasificador robusto.

Puesto que no se pretende desarrollar un sistema de análisis contextual muy complejo, sino demostrar las posibilidades que ofrecen los sensores disponibles, no se ha realizado un estudio más detallado del funcionamiento de los sensores acústicos y se deja éste como sugerencia para futuras ampliaciones.

4.4. Cámara

Finalmente, se ha hecho uso de la cámara integrada en la placa multimedia como mecanismo para el análisis contextual. Al contrario que los sensores acústicos, la cámara no estará activa durante todo el proceso, se utilizará únicamente en caso de que el análisis acústico proporcione un resultado positivo, es decir, en caso de sospecha de una posible caída o accidente.

4.4.1. Descripción General

El chip OV7670 de Omnivision integrado en la placa de sensores IMB400 de Crossbow consta de una cámara con resolucion VGA y un procesador de imagen que permite, en principio, realizar gran parte de las funciones básicas de tratamiento de imagenes, como control de saturación, balance de blanco, ajuste de contraste, etc.

Las principales características se detallan a continuación:

- Alta sensibilidad para el funcionamiento en condiciones de baja iluminación.
- Bajo consumo.
- Soporte de diversos formatos: Raw RGB, RGB565/555/444, YUV 4:2:2, YCbCr 4:2:2.

- Soporte para tamaños de imagen VGA (640x480), CIF (320x240) y escalados[1] hasta 40x30.
- Funciones de control automático de la imagen: ganancia, exposición, balance de blanco y filtrado entre otras.
- Control de la calidad de la imagen: saturación, gamma, contraste, nitidez.
- Función de control de ruido y corrección de defectos.
- Máxima tasa: 30 fps en formato VGA.

4.4.2. Calibración

El análisis de las prestaciones de la cámara se limita a estudiar la calidad de la imagen en los diferentes formatos considerados y el tiempo de transmisión de la imagen completa. Con ello se pretende determinar el formato óptimo para la aplicación de interés

El principal problema que se presenta a la hora de utilizar la cámara está asociado al retardo que se produce en el envío de las imágenes. Teniendo en cuenta que el tamaño máximo de paquete es de 128 bytes, y que una imagen a color en formato VGA ocupa 614.400 bytes, resulta evidente la necesidad de llegar a un compromiso entre el tiempo de transmisión y la calidad de la imagen.

En la Figura 4.25 se muestra la interfaz gráfica de la aplicación desarrollada para el análisis de prestaciones de la cámara. Se puede seleccionar el formato de imagen, así como el retardo introducido en la transmisión y un parámetro de desplazamiento para corregir el que produce la cámara.

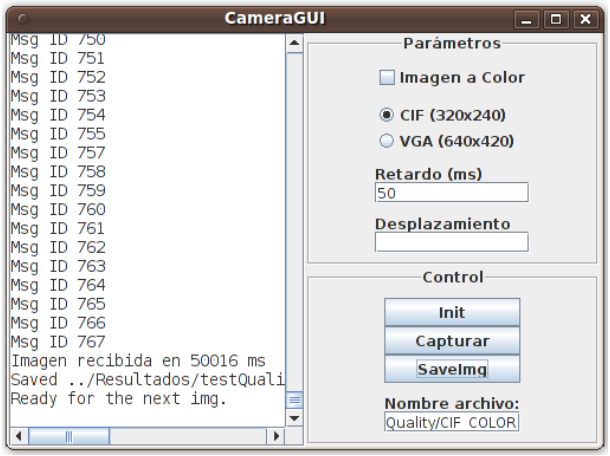


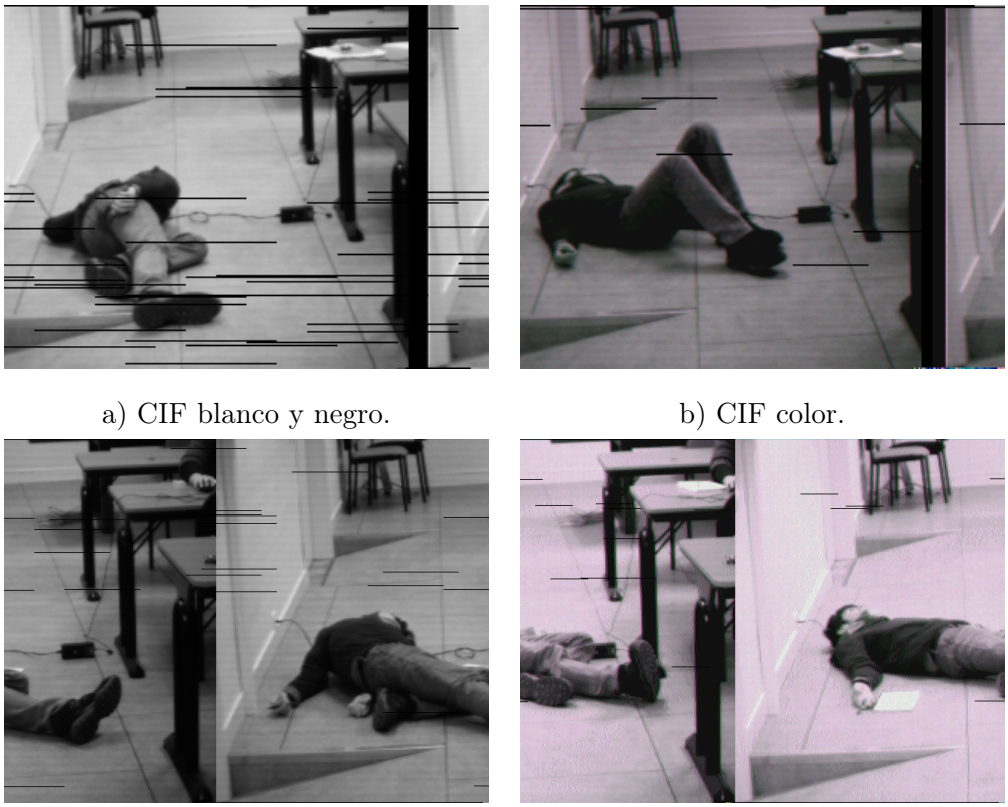
Figura 4.25: GUI para analizar la respuesta de la cámara.

[1]Hasta la fecha no se ha utilizado la capacidad de escalado por problemas en la configuración de la cámara, de modo que sólo pueden utilizarse los tamaños de imagen por defecto.

Calidad de la imagen en función del formato

Para determinar la calidad se tomó una imagen en cada uno de los posibles formatos utilizando una escena representativa de la aplicación para ver su potencial en el análisis contextual. La escena es un aula con una persona que se ha caído y está tumbada en el suelo aparentemente inconsciente.

Actualmente se pueden tomar imágenes en dos tamaños diferentes: CIF (320x240) y VGA (640x480), tanto a color como en blanco y negro. La Figura 4.26 muestra las imágenes obtenidas en los cuatros formatos.



a) CIF blanco y negro. b) CIF color.

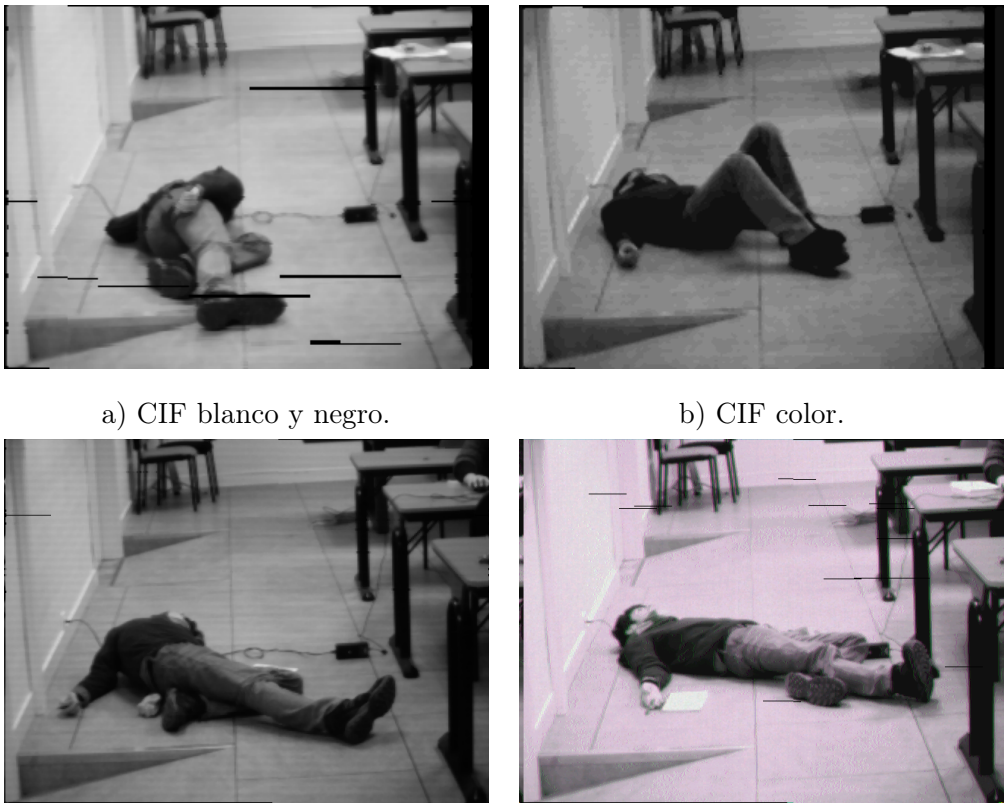
c) VGA blanco y negro. d) VGA color.

Figura 4.26: Imágenes en diferentes formatos

De la Figura 4.26 se desprende que las imágenes supuestamente a color no representan bien los colores. Esto se debe a un problema en la configuración de la cámara, por lo que en principio los formatos a color quedan descartados, además, ocupan el doble de espacio que las imágenes en blanco y negro y por tanto el tiempo de transmisión será también el doble. En cuanto a la diferencia entre utilizar CIF o VGA, la mejora de calidad en VGA no resulta tan significativa como para justificar el aumento del tiempo de transmisión que supondría, ya que una imagen en

blanco y negro en formato CIF ocupa 76.800 bytes, mientras que en VGA serán 307.200 bytes, cuatro veces más. Por tanto, el formato por defecto será siempre CIF en blanco y negro.

Por otra parte, en las imágenes mostradas se observa un desplazamiento de la imagen, que aparenta estar desordenada. También se pueden observar los errores de transmisión que se producen, que aparecen como líneas negras en distintas zonas de las imágenes. Los errores de transmisión se discuten en más detalle en el siguiente apartado, donde se justificará que la imagen pueda presentar algunos de estos errores. En cualquier caso, utilizando un sencillo procesado, se puede mejorar la calidad y reordenar la imagen. Para ello basta con desplazar una cierta cantidad la imagen en el eje horizontal y aplicar un filtro de mediana para eliminar la mayor parte de los errores de transmisión. El resultado de aplicar este proceso a las imágenes anteriores se muestra en la Figura 4.27.



a) CIF blanco y negro. b) CIF color.

c) VGA blanco y negro. d) VGA color.

Figura 4.27: Imágenes en diferentes formatos modificadas.

Estudio del tiempo de transmisión

Una vez fijado el formato, es necesario resolver un problema que se presenta a la hora de transmitir las imágenes: la pérdida de paquetes, y por tanto de fragmentos de imagen. Esta pérdida se debe al limitado tamaño del buffer del receptor, de 128 bytes, equivalente a una única trama, de manera que si el tiempo de transmisión de una trama es menor que el tiempo que se tarda en procesarla y vaciar el buffer de recepción, se descartarán los paquetes recibidos que no se puedan almacenar.

Antes de analizar las imágenes y los tiempos obtenidos experimentalmente, se realiza un análisis teórico del tiempo de transmisión de una imagen en diferentes formatos.

Teniendo en cuenta que no se utilizan direcciones extendidas y que el tamaño máximo de una trama a nivel físico es de 128 bytes, se pueden transmitir 115 bytes de datos útiles en cada trama. Si se restan los bytes de cabeceras a nivel de aplicación[2] que se han definido, cada trama llevará como máximo 100 bytes de datos de la imagen.

Por otra parte, la tasa de transmisión es R = 250 kbps y por tanto el tiempo de transmisión de un paquete en función de su longitud, L, será:

$$t_{tx} = \frac{L}{R} \Rightarrow \text{máx}\, t_{tx} = \frac{128 \cdot 8}{250 \cdot 1000} = 4{,}1\, ms \tag{4.4}$$

En la Tabla 4.4 se muestra el tiempo de transmisión teórico de una imagen en formatos CIF y VGA en blanco y negro (1 byte/pixel). Los valores reflejados en la Tabla 4.4 corresponden al tiempo que tarda la mota en transmitir la imagen completa sin considerar tiempos de procesamiento y considerando el retardo de propagación nulo. Esta última suposición resulta razonable puesto que la distancia entre el nodo con la cámara y la estación base que hace de interfaz no será muy grande ($< 10m$), por lo que el retardo de propagación será inferior a $1\mu s$. El tiempo de procesamiento sin embargo sí será relevante para el retardo real de transmisión.

$$N = \frac{NTotalbytes}{Max.bytesportrama} \Rightarrow t_{total} = N \cdot t_{tx} \tag{4.5}$$

Tabla 4.4: Tiempos de transmisión teóricos.

Formato	**Tamaño (bytes)**	**Tiempo de transmisión (sg)**
CIF	76800	3.1
VGA	307200	12.6

[2]Considerando el formato de trama utilizado en la aplicación final.

A priori parece que los tiempos de transmisión son razonables, sin embargo, si se observa la imagen recibida (Figura 4.28) el resultado refleja la pérdida de la mayor parte de la imagen durante la transmisión.

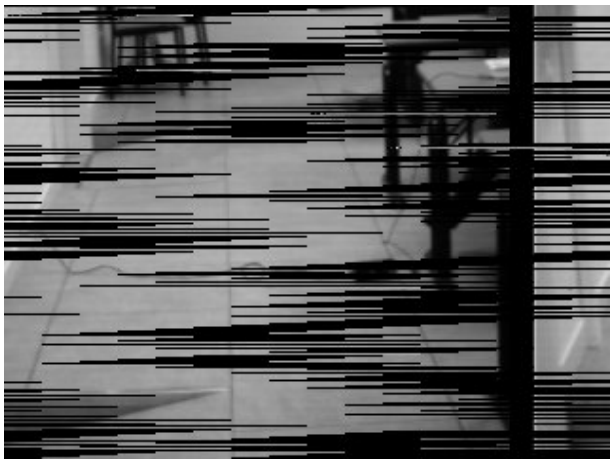


Figura 4.28: Imagen inicial en formato CIF en blanco y negro.

El problema radica en el hecho de que el buffer del receptor está limitado a 128 bytes, es decir, a una única trama. El tiempo que tarda el receptor en procesar la trama y vaciar el buffer es mayor que el tiempo de transmisión, de manera que se descartan paquetes. Este hecho hace necesaria la introducción de un retardo en la transmisión o bien de la implementación de un sistema de control de flujo que permita gestionar la pérdida de paquetes.

Se ha optado por la solución más sencilla con el fin de reducir la complejidad de la aplicación. De este modo, el transmisor espera un cierto tiempo t_{delay} antes de cada envío. Este retardo se ha fijado experimentalmente y se ha llegado a un compromiso admitiendo que puedan perderse algunos paquetes sin que la imagen se vea severamente afectada.

La Tabla 4.5 muestra, para una serie de valores de t_{delay}, el tiempo total que se tarda en recibir la imagen completa, desde que se da la orden de captura de la imagen hasta que se recibe el último paquete, y el número de paquetes que se han perdido. Se tomaron dos imágenes para cada retardo con formato CIF en blanco y negro, puesto que será el que se utilice por defecto. Los valores presentados son la media de los obtenidos en ambas imágenes.

Los valores de la Tabla 4.5 se muestran de forma gráfica en la Figura 4.29. De ella se puede concluir que el valor óptimo de retardo está entre 30 y 40 ms, puesto que el porcentaje de pérdidas es bajo y el tiempo de transmisión es de poco más de medio minuto.

Tabla 4.5: Tiempo de recepción de una imagen completa.

$t_{delay}[ms]$	$t_{tx}^{tot}(ms)$	**Paquetes perdidos (%)**
0	22.81	35.61
10	30.15	17.58
20	31.27	3.19
30	35.06	2.21
40	42.54	1.89
50	50.62	1.69
60	57.90	2.08
70	65.65	2.60
80	73.48	2.08

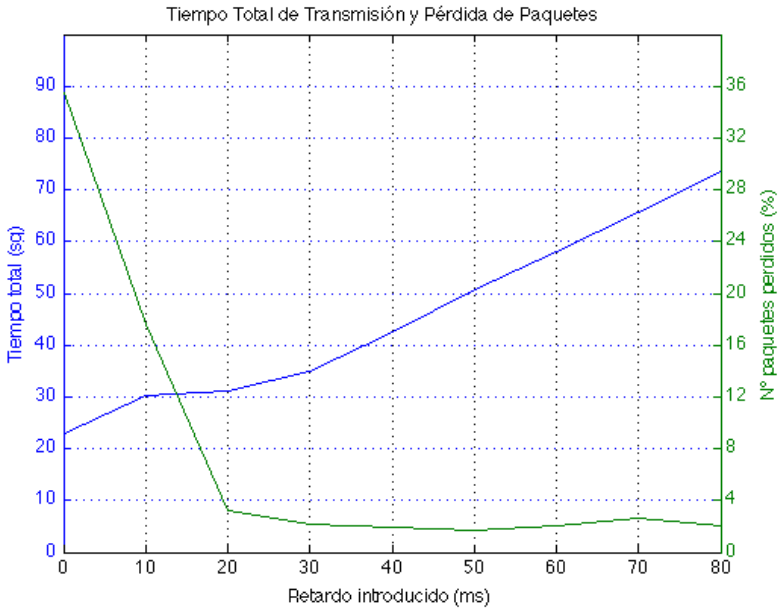


Figura 4.29: Tiempo total de transmisión y porcentaje de paquetes perdidos para distintos retardos.

Un aspecto importante a tener en cuenta es la orientación de la mota que recibirá la imagen. Según se ha podido comprobar experimentalmente, si se sitúa la mota en perpendicular al suelo (por ejemplo, pegada a una pared), las pérdidas de paquetes son sustancialmente mayores que si se sitúa paralela al suelo. La Tabla 4.6 muestra el porcentaje de paquetes perdidos para cada una de las orientaciones y para los retardos considerados anteriormente.

Tabla 4.6: Porcentaje de pérdidas según la orientación de la mota

$t_{delay}[ms]$	**Orientación Vertical**	**Orientación Horizontal**
0	41.41	35.61
10	16.86	17.58
20	13.15	3.19
30	16.28	2.21
40	8.40	1.89
50	14.45	1.69
60	12.43	2.08
70	12.24	2.60
80	13.61	2.08

Se puede observar que los valores en orientación vertical distan mucho de los que se pueden alcanzar con la orientación contraria, de manera que para la aplicación final será importante tener en cuenta este dato a la hora de distribuir los sensores.

Capítulo 5

Descripción de la Solución Propuesta

En este capítulo se describe la solución propuesta para el problema de la detección y seguimiento de personas en entornos *indoor* utilizando la tecnología Imote2.

La solución se puede descomponer en dos grandes fases: la primera consiste en implementar el algoritmo de detección y seguimiento y la segunda en añadir capacidad de análisis contextual al sistema. Para la primera fase se utilizaron sensores de movimiento y lumínicos, mientras que para la segunda se emplearon los micrófonos y el códec de audio combinado con la cámara.

Antes de pasar a describir la aplicación, es conveniente recordar una restricción fundamental del problema: se asume que únicamente puede haber una persona en la zona de despliegue de la aplicación, en el momento en que esta restricción no se cumpla dejará de funcionar correctamente.

5.1. Descripción General

Para poder explicar el funcionamiento de la aplicación, en primer lugar se van a describir los diferentes tipos de nodos que van a formar parte de la misma.

Gateway: Es el nodo que hace de enlace entre la red y el ordenador que permite controlar la aplicación y que recibe los resultados.

Nodos raíz (*rootNode*): Uno por cada subsistema de detección. Cada nodo raíz controlará los nodos sensores de su zona, y hará de enlace entre éstos y la interfaz, además de comunicarse con el resto de nodos raíz para notificar eventos.

Nodos con sensor de movimiento (*pirSensor*): Uno por cada subsistema de detección. Alertarán de la entrada de la persona en su área, comunicándolo a su nodo raíz para que tome las decisiones oportunas.

Nodos con sensores lumínicos (*lightSensor*): Se desplegarán varios en un mismo área. Servirán para realizar una localización más precisa dentro un área.

Nodos con capacidad multimedia (*audioCameraSensor*) : Podrá haber uno o varios en cada área en función de su amplitud. Su función será obtener información acústica y determinar si se ha producido alguna situación anómala, como pueda ser una caída de una persona. Si esto ocurre se tomará una imagen y se enviará al ordenador central para que sea procesada.

El desarrollo de la aplicación puede descomponerse en varias fases, partiendo de un problema más sencillo al que se va añadiendo funcionalidad. Así, se pueden distinguir las siguientes etapas de desarrollo:

1. Desarrollo e implementación del algoritmo de detección de los sensores lumínicos. Este algoritmo debe detectar la presencia de una persona en base a una variación de iluminación fijada por un umbral de detección. Además, deberá tener en cuenta la posibilidad de que se enciendan o apaguen las luces y deberá determinar si la persona se está acercando o alejando del sensor.

2. Desarrollo del sistema básico de análisis contextual. En esta fase se desarrolla el sistema de análisis contextual básico que se va a utilizar. Con la información recogida por los micrófonos se puede establecer si se ha producido un evento atípico, en cuyo caso se utilizará la cámara para enviar las imágenes al ordenador central.

3. Desarrollo de los subsistemas. En esta etapa se integran los nodos *lightSensor*, con el algoritmo de detección desarrollado, los nodos encargados del análisis contextual, *audioCameraSensor*, el sensor de movimiento *pirSensor* y el *rootNode*. Con esta etapa se quiere conseguir que el *pirSensor* reaccione ante la detección de movimiento despertando al resto de motas que previamente habían entrado en un estado de bajo consumo. Este mecanismo busca gestionar la energía de manera que sólo estén activos los nodos del área en el que se encuentra la persona.

4. Desarrollo final. Finalmente se integran los diferentes subsistemas desarrollando el protocolo de comunicación entre los nodos raíz de manera que al detectar presencia en un área, el *rootNode* responsable de la misma comunique al resto de nodos raíz el evento para que puedan ordenar a sus respectivos nodos sensores la entrada en modo de bajo consumo.

La forma en que interactúan los diferentes subsistemas y sus nodos se resume de forma gráfica en la Figura 5.1. En ella se puede ver la estructura jerárquica, en la que se puede controlar el funcionamiento a través de un ordenador, que interactúa con la red a través del *gateway*. Los nodos raíz forman el segundo nivel, recibiendo órdenes y controlando sus respectivos nodos sensores. Finalmente los sensores forman el último nivel, cada uno con una función de acuerdo a su tipo.

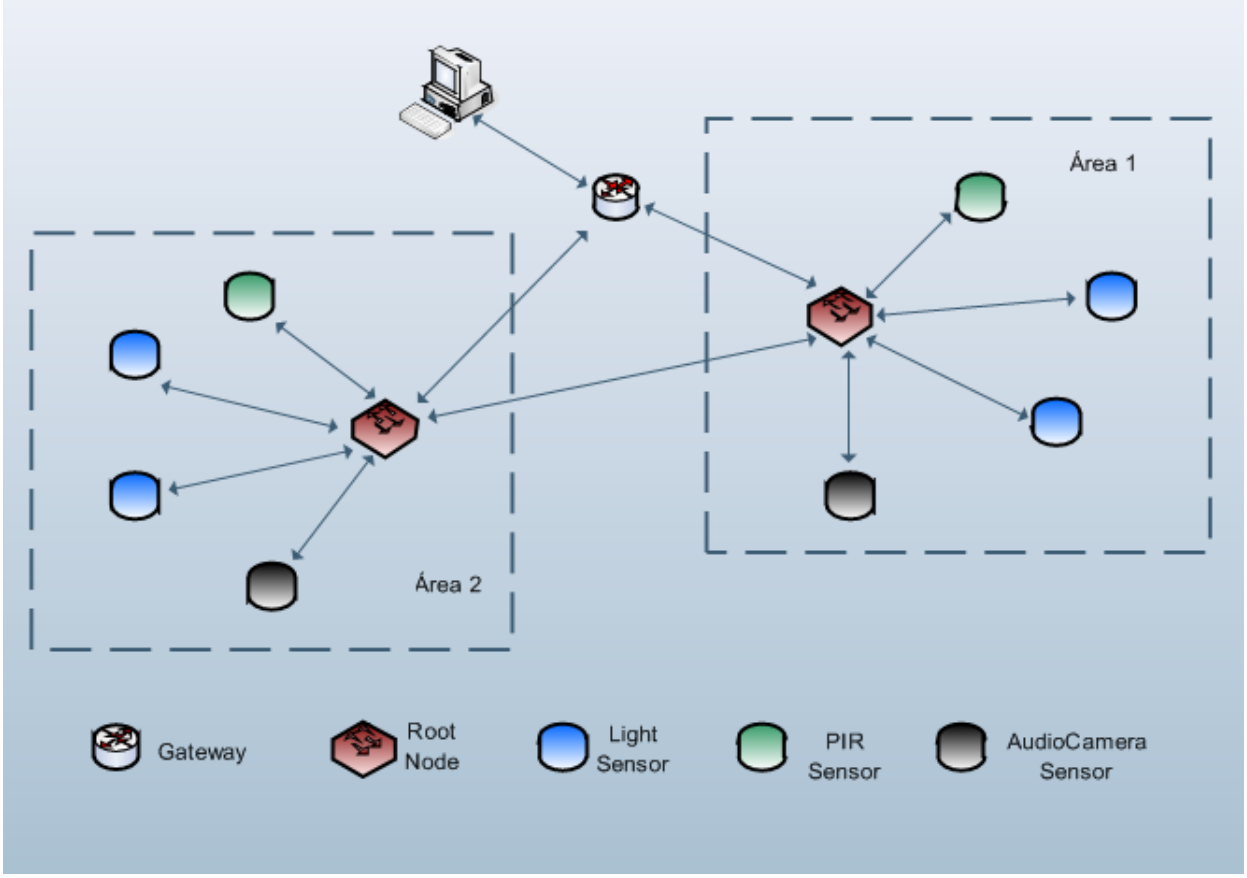


Figura 5.1: Arquitectura de la aplicación.

En los siguientes apartados se describe cada una de las etapas de forma detallada.

5.2. Etapa 1: Algoritmo de Detección

Para poder desarrollar el algoritmo de detección, es importante tener en cuenta la calibración de los sensores lumínicos descrita en el Capítulo 4, en concreto el apartado referente al umbral de detección. Puesto que el sistema debe funcionar en diferentes condiciones de iluminación y

éstas van variando a lo largo del día, el umbral no puede ser un valor de iluminación estático, debe ser un porcentaje de variación con respecto a un valor de referencia que irá cambiando a lo largo del tiempo. Este valor de referencia será la media de una cantidad prefijada de medidas del sensor. De esta forma, la media va cambiando a medida que se toman valores, y el umbral de detección se va actualizando en consecuencia. Esta actualización únicamente tiene lugar si se ha determinado que no existe presencia en el rango de detección del sensor.

El algoritmo parte de los siguientes parámetros:

- α: Porcentaje de variación respecto a la media para considerar que se ha detectado presencia. Permite fijar el umbral de detección.
- β: Porcentaje de variación respecto a la media para considerar que se han encendido/apagado las luces.
- N: Número de muestras con las que se calcula la media.
- $samplePeriod$: Periodo de muestreo del sensor.
- $fastSamplePeriod$: Periodo de muestreo más rápido que se utiliza al iniciar el sensor.
- $gain$: Determina si se utiliza la ganancia del sensor lumínico o no.

Además, se utilizan una serie de variables que se describen a continuación para poder entender mejor el pseudo-código incluido.

- $meanArray$: Array que almacena las últimas N medidas del sensor y que sirve para calcular la media
- $memoria$: Array que se utiliza cuando se detecta presencia para almacenar las 2 últimas medidas. El objetivo es determinar si la persona se está acercando o alejando del sensor. Para ello se compara el valor medido con el valor dos instantes antes. El motivo de no comparar con el inmediatamente anterior es evitar errores debidos a la varianza de las medidas, se considera que saltando una muestra la diferencia es suficientemente significativa como para dar un resultado fiable. Por otra parte, el hecho de requerir una memoria de dos posiciones supone que en dos instantes el movimiento será desconocido.
- $lastResult$: Almacena el resultado de la etapa de detección anterior.

Con los parámetros y variables ya definidos, entender el algoritmo resulta más sencillo. Junto con la descrición del algortimo se incluye el pseudocódigo asociado para que resulte más sencillo de entender.

1. **Inicialización**: Se utiliza un periodo de muestreo (*fastSamplePeriod*) mucho menor que el normal para obtener las N muestras necesarias para calcular la media de forma rápida y así poder pasar a la etapa de detección reduciendo el retardo inicial.

```
while i < N {
 m = leerSensor()
 if m !=  0 {
        meanArray[i]  =  m
        i++
     }
 wait (fastSamplePeriod)
}
media = mean(meanArray)
```

2. **Detección**: Una vez calculada la media inicial, se pasa a la fase de detección, en la que el sensor toma una medida cada *samplePeriod* ms. Se compara la medida con los umbrales de detección para decidir el resultado: detección, encendido de luces, apagado de luces o no detección.

   Calcular Umbrales:

```
detectLimit = media (1 - alpha/100)
lightsOnLimit = media (1 + beta/100)
lightsOffLimit = media (1 - beta/100)
```

Leer el sensor y decicir el resultado.

```
m = leerSensor
if m > lightsOnLimit
    resultado = LIGHTS_ON
else if m < lightsOffLimit
    resultado = LIGHTS_OFF
else if m < detectLimit
    resultado = DETECT
else
    resultado = NO_DETECT
```

3. **Actuación**: Tras decidir, se pasa a actuar de acuerdo con la decisión tomada. En caso de que el resultado sea de no-detección, simplemente se almacena la muestra en el array de medias y se recalcula la media y los umbrales. Si se ha decido detección, es necesario determinar el tipo de movimiento de la persona: acercándose al sensor, alejándose o desconocido si son las dos primeras muestras para las que se decide detección. Finalmente, si el resultado es encendido o apagado de luces se para el temporizador, se apaga y se reconfigura el sensor desactivando/activando la ganancia y se vuelve a iniciar el algoritmo calculando de nuevo la media inicial.

```
if resultado = NO_DETECT
    meanArray [inPtr] = m
    inPtr = ( inPtr + 1 ) % N
    mean = mean( meanArray )
    lastResult = NO_DETECT
else if resultado = DETECT
    if  lastResult != DETECT
        memory[0] = 0
        memory[1] = 0
        movimiento = UNKNOWN
    else if memory[0] == 0
```

```
            movimiento = UNKNOWN
        else if m <= memory[0]
            movimiento = ACERCANDOSE
        else
            movimiento = ALEJANDOSE
        memory[0] = memory[1]
        memory[1] = m
        lastResult = DETECT
        notificar
    else if resultado = LIGHTS_ON
        parar timer
        apagar sensor
        ganancia = FALSE
        lastResult = LIGHTS_ON
        reiniciar sensor
        notificar
    else if resultado = LIGHTS_OFF
        parar timer
        apagar sensor
        ganancia = TRUE
        lastResult = LIGHTS_OFF
        reiniciar sensor
        notificar
```

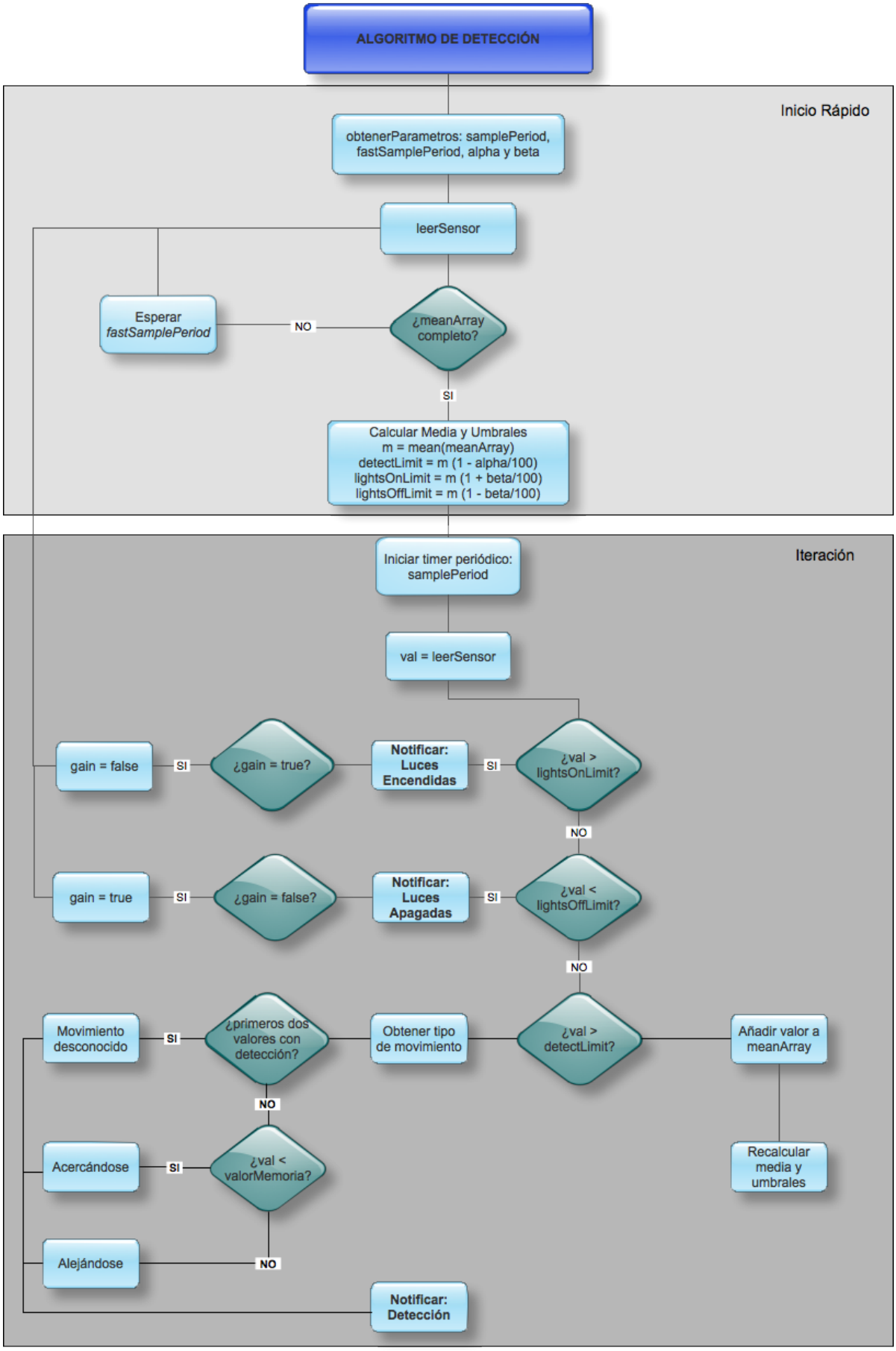


Figura 5.2: Diagrama de flujo del algoritmo de detección.

Cómo sistema para saber lo que está ocurriendo, se han utilizado los LEDs de la placa, de manera que la equivalencia entre los colores y lo que sucede es la siguiente:

- Verde: En funcionamiento. Midiendo cada *samplePeriod* ms.
- Rojo: Presencia detectada.
- Azul: Inicio rápido o encendido/apagado de luces durante el funcionamiento normal.

5.3. Etapa 2: Análisis Contextual

El sistema implementado utiliza la información recogida por el micrófono integrado en la placa multimedia para decidir si ha ocurrido un evento anómalo, en cuyo caso se toma una imagen de la escena que se envía al ordenador central para analizarla.

La complejidad de esta etapa reside en cómo utilizar la información del micrófono y qué criterio seguir a la hora de decidir. Como se explicó en el Capítulo 4, en el apartado referente a los sensores acústicos, se puede establecer un umbral a partir de la energía de la señal de audio recibida. De este modo, el funcionamiento es el siguiente: el nodo multimedia comienza a grabar ventanas de *winlen* ms. De cada ventana se calcula la *Short Time Energy, STE*, y se compara con el umbral de detección fijado. Si la energía de la ventana supera el umbral, se toma una imagen y se envía, dejando de grabar durante el tiempo que tarda la transmisión. Una vez la imagen se ha enviado, se inicia de nuevo la grabación y se repite el proceso. El diagrama de flujo del algoritmo se puede ver en la Figura 5.3.

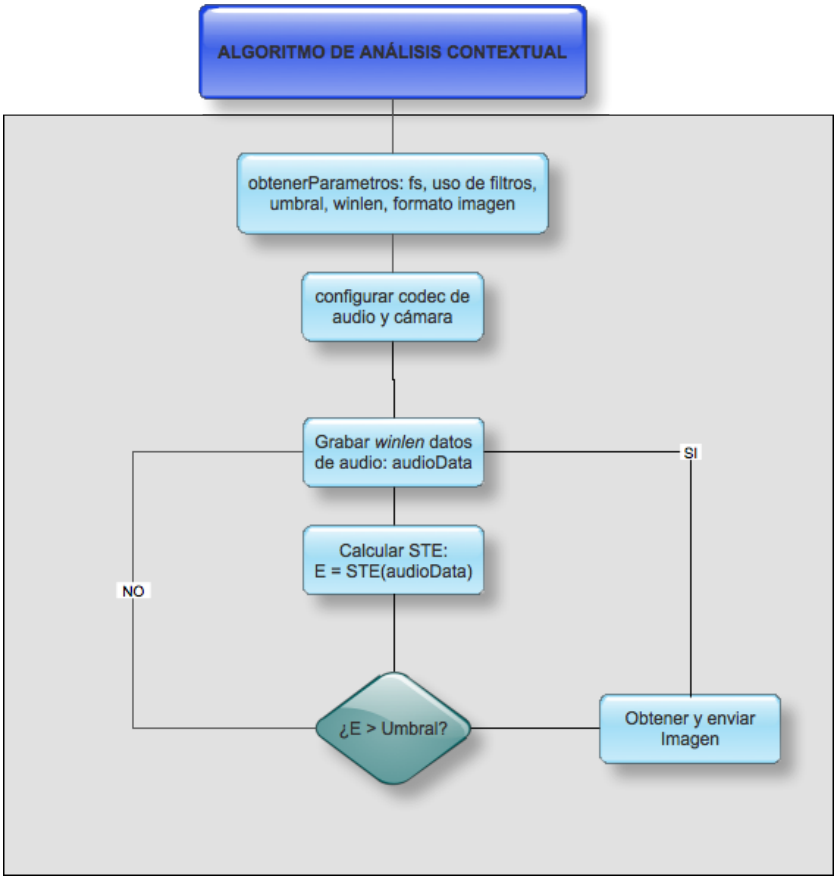


Figura 5.3: Diagrama de flujo del algoritmo de análisis contextual.

5.4. Etapa 3: Desarrollo de un Subsistema

Como se puede ver en la Figura 5.1, cada subsistema cuenta con un nodo raíz, un nodo con sensor de movimiento, varios nodos con sensores lumínicos y un nodo con capacidad multimedia. En esta sección se describe el funcionamiento interno de esos subsistemas, detallando la comunicación entre los diferentes tipos de nodos.

Inicialmente, todos los nodos se encuentran en un estado de bajo consumo, de manera que sólo se activan ante determinados eventos. Para controlar cuándo deben activarse los nodos de un subsistema se utilizan los sensores de movimiento. Se parte por tanto de un subsistema en estado de bajo consumo en el que todos los nodos, incluido el nodo raíz, se encuentran inactivos. Cuando el sensor de movimiento detecta la presencia de una persona, genera una interrupción que activa la mota y permite enviar un mensaje de notificación a su nodo raíz. Éste, ante la recepción del mensaje se activa y envía la orden para que los sensores empiecen a ejecutar el algoritmo de detección.

A partir de este momento los nodos con sensores lumínicos ejecutan el algoritmo de detección enviando notificaciones al nodo raíz únicamente cuando se decide que se ha detectado presencia o encendido/apagado de luces. Cuando no se detecta presencia no se envía ningún mensaje para reducir el tráfico y ahorrar energía, ya que el consumo al utilizar la radio es muy alto. Por otra parte, los sensores multimedia ejecutan su propio algoritmo comunicándose con el nodo raíz sólo cuando se detecta un evento anómalo, en cuyo caso se envía la imagen tomada por la cámara. En caso de que esto ocurra, los mensajes con la imagen tendrán prioridad sobre los enviados por los *lightSensor*.

Para entender la comunicación entre los diferentes elementos del subsistema se deben identificar primero los tipos de mensajes que se van a transmitir y entre qué nodos se dará esa comunicación. Estos mensajes se describen a continuación.

***CmdMsg*:** Mensaje de control utilizado para indicar qué acción se debe realizar y cuáles son los parámetros de funcionamiento. Se pueden generar desde el ordenador central a los nodos raíz, o desde éstos a los nodos sensores.

Comandos del ordenador central a los nodos raíz:

- HELLO: Simplemente sirve para que las motas contesten con su identificador. Ante la recepción de un comando HELLO, cada nodo raíz reenvía el mensaje a sus nodos sensores y espera a recibir respuestas antes de contestar.
- START: Inicia la aplicación con los parámetros fijados a través de la interfaz gráfica. Este mensaje no se reenvía, en cambio se genera otro *CmdMsg* con el comando PIR_START_CMD, al que sólo atenderá el sensor de movimiento. De esta forma, se indica al *pirSensor* que debe empezar a atender las interrupciones y por tanto a detectar la presencia de una persona. El resto de sensores permanece en modo de bajo consumo a la espera de que se detecte presencia.
- STOP: Detiene la aplicación, se reenvía directamente a los nodos sensores.

Comandos de los nodos raíz a los sensores:

- HELLO: Cumple la misma función que el recibido por el nodo raíz, al recibirlo los sensores contestan con su identificador.
- PIR_START_CMD: Comando específico para los nodos con sensor de movimiento. Al recibirlo el nodo sensor empieza a atender las interrupciones generadas al detectar presencia.

- START: Se genera en el nodo raíz cuando se ha detectado presencia y se envía a los nodos sensores para que salgan del estado de bajo consumo y comiencen a ejecutar los algoritmos de detección que les correspondan.

- STOP: Detiene la aplicación haciendo que los sensores entren en modo de bajo consumo.

***HelloMsg*:** Mensaje de HELLO de los nodos sensores a su nodo raíz.

***RootHelloMsg*:** Mensaje de HELLO de un nodo raíz al gateway. En él se combinan los mensajes de HELLO enviados por todos los sensores del área.

***PirMsg*:** Mensaje del *pirSensor* al nodo raíz indicando que se ha detectado presencia.

***LightSensorMsg*:** Mensaje de un nodo *lightSensor* al nodo raíz con los resultados de detección. Incluye datos como la media y los umbrales de detección. El nodo raíz reenvía el mensaje al ordenador central.

***ImgMsg*:** Mensaje enviado por el *audioCameraSensor* a su nodo raíz con parte de la imagen capturada y reenviado por éste al ordenador central.

La Figura 5.4 muestra la relación entre los diferentes mensajes y comandos posibles.

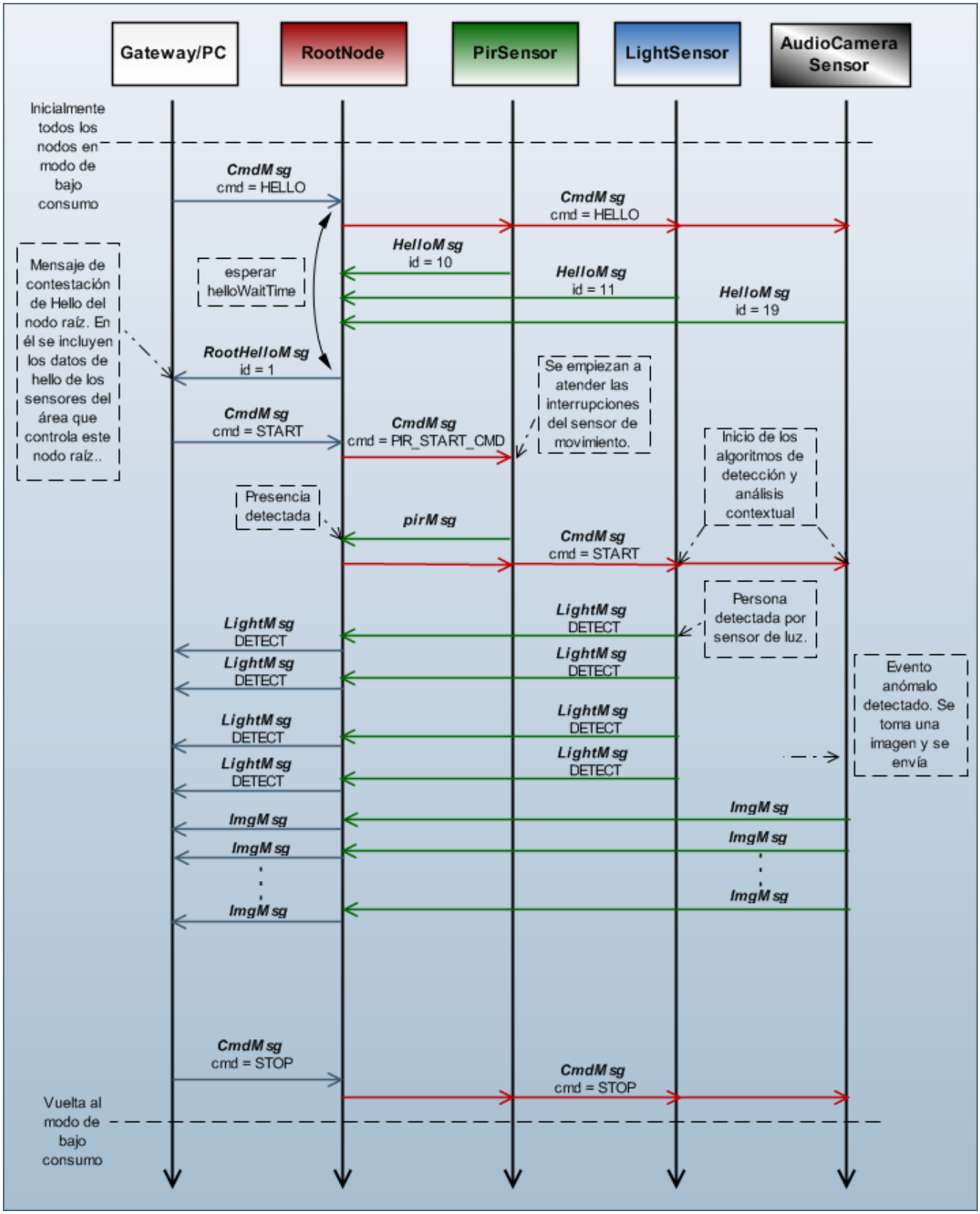


Figura 5.4: Diagrama del intercambio de mensajes dentro de un subsistema.

5.5. Etapa 4: Sistema Final

Con los subsistemas ya funcionando, sólo queda determinar la forma de comunicación entre subsistemas y cómo se gestiona la comunicación entre los distintos niveles de la jerarquía.

Se pueden establecer dos canales diferentes: el primero permite el intercambio de mensajes entre los nodos raíz y entre éstos y el *gateway* y el segundo permite que un nodo raíz se comunique sólo con los sensores de su área. A los tipos de mensajes descritos anteriormente se añade uno nuevo de comunicación entre nodos raíz.

***RootNotifyMsg*:** Mensaje de notificación que envía un nodo raíz al resto y al *gateway* informando de la presencia de la persona en su área.

Con los mensajes ya definidos, la secuencia de funcionamiento del sistema es la siguiente: se parte de un estado en el que todas las motas están inactivas. A través de la interfaz de usuario se puede solicitar un mensaje de hello para conocer los nodos existentes. El comando HELLO enviado a través del *gateway* lo reciben únicamente los nodos raíz, que a su vez envían un *CmdMsg* con el mismo comando a sus nodos sensores. Éstos contestan a su *rootNode*, que tras un tiempo esperando recibir mensajes compone uno nuevo combinando los mensajes recibidos y enviándolo al ordenador central. Los mensajes de HELLO no hacen que la apliación empiece a funcionar, simplemente permiten comprobar que los nodos utilizados funcionan correctamente.

El funcionamiento del sistema comienza con el envío del comando START desde el ordenador a los nodos raíz. Todas las motas se encuentran inactivas en el momento de enviar este comando. Al recibir el mensaje, cada *rootNode* envía un *CmdMsg* con el comando PIR_START_CMD dirigido al nodo con sensor de movimiento para que empiece a atender las interrupciones generadas al detectar presencia. Esto se realiza cambiando el estado del nodo, que pasa de un estado IDLE a un estado que se ha denominado WAIT_ENTRY. Las interrupciones solamente se atienden cuando la mota se encuentra en este estado. Tras esta etapa todos los subsistemas se encuentran esperando detectar la presencia de la persona en su área para comenzar a funcionar.

Cuando un sensor de movimiento detecta presencia, envía un mensaje tipo *PirMsg* a su nodo raíz. Éste inmediatamente crea un *CmdMsg* con el comando START dirigido al resto de nodos sensores para que inicien el algoritmo de detección. Por otra parte, envía una notificación *RootNotifyMsg* al resto de nodos raíz y al ordenador informando de la presencia de la persona en su área. Al recibir este mensaje, el *rootNode* del subsistema que estaba activo ordena a sus sensores detener el algoritmo y entrar en estado inactivo esperando de nuevo detectar presencia en su área.

La Figura 5.5 muestra la secuencia de intercambio de mensajes entre los nodos raíz y el *gateway* en un sistema compuesto por dos subsistemas. El funcionamiento del sistema cuando un nodo raíz recibe un comando del ordenador es el mostrado en la Figura 5.4.

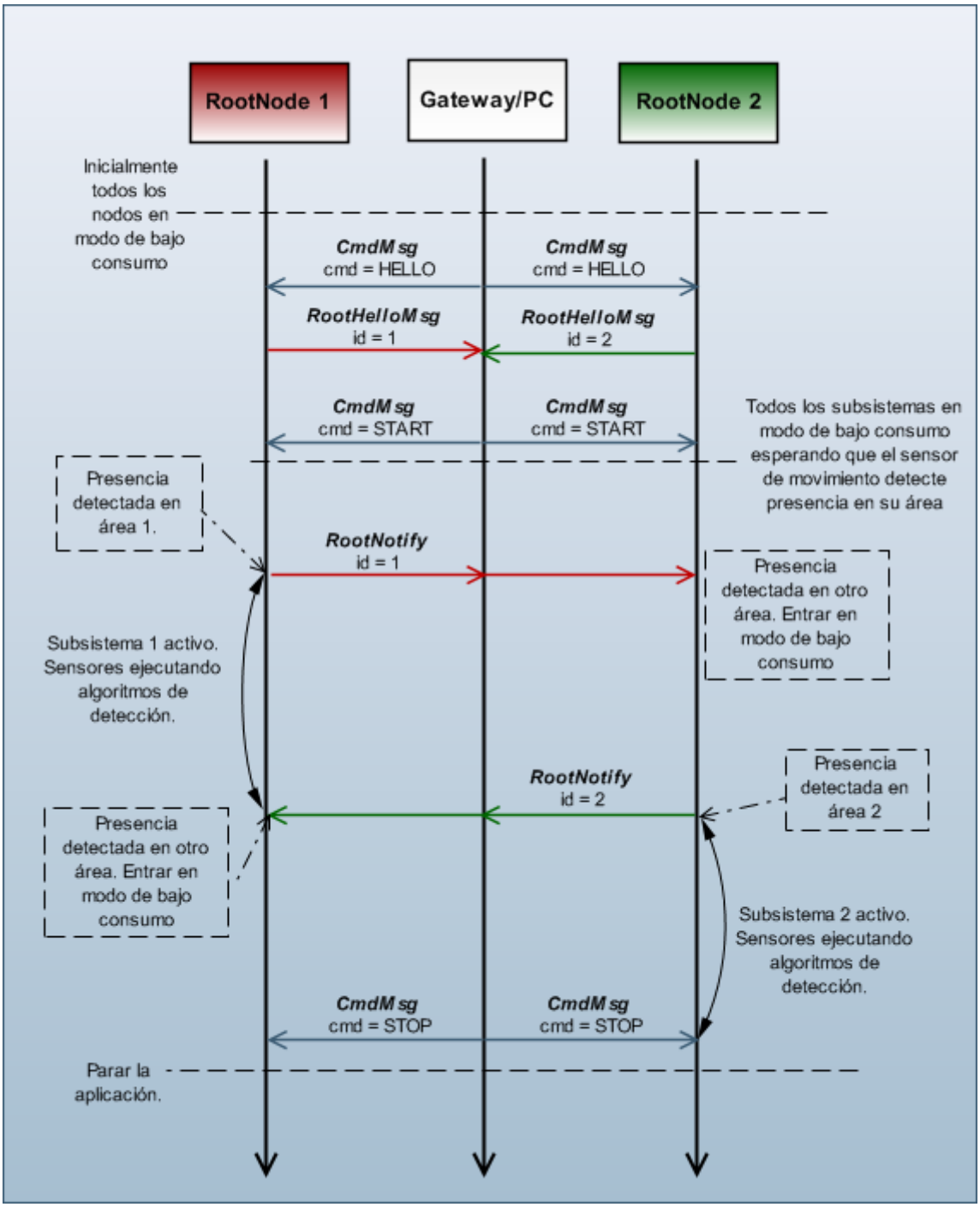


Figura 5.5: Diagrama del intercambio de mensajes entre nodos raíz y el ordenador.

El ordenador central irá registrando los mensajes recibidos, generando un archivo en el que se especifica dónde y a qué hora se ha detectado la presencia de la persona. También almacena los datos de los sensores que han detectado la presencia: valor medido, decisión, umbrales de detección, tipo de movimiento, etc. Además, en caso de que se produzca un evento anómalo de acuerdo con el análisis contextual, se almacenará la imagen recibida.

Un aspecto crítico de la aplicación es la forma de enviar los mensajes para asegurar que un nodo raíz recibe únicamente mensajes de sensores de su área y éstos sólo reciben mensajes del *rootNode* que les corresponde. Para ello se ha utilizado el siguiente convenio a la hora de asignar un identificadores a las motas:

***Gateway*:** Siempre será el identificador 0.

***RootNode*:** Un número comprendido entre 1-9 (asumiendo que no habrá más de nueve estancias diferentes en la casa).

***PirSensor, LightSensor y AudioCameraSensor*:** Identificador de dos dígitos, siendo el primero igual al del nodo raíz de su área. Por ejemplo, si el nodo raíz tiene el identificador 1, todos los sensores de su área tendrán identificadores entre 10 y 19. Generalmente el primer valor se reserva para el sensor de movimiento y el último para el sensor multimedia.

Así, se asigna a cada nodo un identificador de grupo además del suyo propio. Para los nodos raíz el identificador de grupo será siempre el 0, mientras que para los nodos sensores será el identificador del nodo raíz responsable del área en el que se encuentre. Con este sistema, cada mensaje que se envía contiene el identificador del nodo que lo envía y el identificador del grupo al que está destinado, de manera que se puedan descartar los mensajes recibidos cuyo grupo no se corresponda con el propio del nodo.

5.6. Consideraciones Sobre Potencia

La potencia de transmisión de las motas será un parámetro de diseño que se fija con el fin de optimizar la energía, de manera que cada mota debe emitir con la mínima potencia para que el paquete llegue a su destino. De acuerdo con las hojas de catálogo, se puede variar dicha potencia entre 0 y -24 dBm. En la Tabla 5.1 se muestra la relación entre la potencia de transmisión y el consumo de corriente asociado únicamente a la radio.

Tabla 5.1: Corriente consumida para distintas potencias de transmisión.

Potencia Tx. [dBm]	**Corriente consumida [mA]**
-25	8,5
-15	9,9
-10	11
-5	14
0	17,4

Por otra parte, los datos de consumo de la radio en los diferentes modos demuestran por si mismos la necesidad de utilizar una política de ahorro energético en las motas (Tabla 5.2); en modo de inactividad se reduce el gasto energético un 95 % si se entra en el modo de bajo consumo.

Tabla 5.2: Corriente consumida según el modo de funcionamiento.

Modo	**Corriente consumida**
Bajo consumo (*power down*)	20μA
Idle	426μA
Recibiendo	18,8 mA

5.7. Interfaz de Usuario

El desarrollo de la interfaz de usuario se realizó utilizando java. Esta elección se debe a las facilidades que proporciona tanto para la creación de GUI como para la comunicación con redes externas, en este caso con la red de sensores a través de un puerto usb.

Para la gestión de los componentes de la GUI y su creación se ha utilizado el paquete *javax*, siendo el resultado el mostrado en la Figura 5.6.

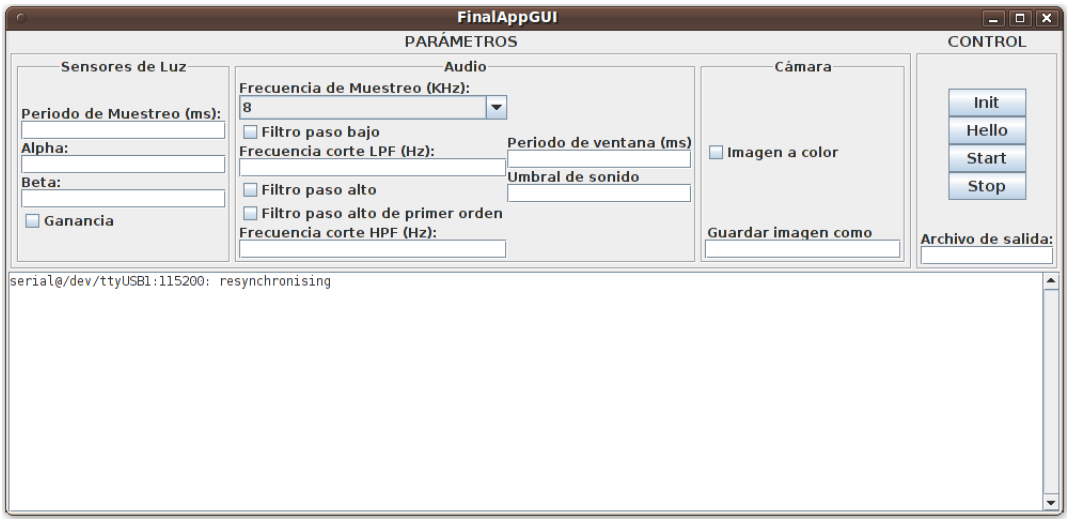


Figura 5.6: Interfaz de usuario de la aplicación completa.

Una vez diseñada la interfaz se añadió la funcionalidad deseada. Para ello se realizaron las siguientes funiciones:

1. Obtención de parámetros. Interpreta los parámetros introducidos por el usuario y comprueba que tienen valores correctos. Si no se ha introducido ningún valor se asigna el valor por defecto.

2. Creación de fichero de salida. Crear y añadir cabeceras del fichero para que esté listo cuando empiecen a llegar datos.

3. Recepción de mensajes de la red. Interpreta los paquetes recibidos. Si se recibe un mensaje *rootHello* se imprimen los datos en la pantalla. Si es un mensaje *rootNotify* significa que el usuario ha cambiado de zona, por lo que se añaden unas líneas de separación en el archivo de salida y se indica la zona en la que se ha detectado y la hora a la que ha ocurrido. Si es un mensaje de un sensor de luz, simplemente se obtienen los datos y se añaden al fichero. Finalmente, si es un paquete tipo *imgMsg* se almacenan los datos de la imagen. Si el paquete contiene los últimos bytes de la imagen, se crea un archivo con las cabeceras adecuadas y el formato necesario para guardar la imagen correctamente.

4. Envío de mensajes. Al pulsar cualquiera de los botones de control, se obtienen los parámetros y se crea el *cmdMsg* que se va a enviar.

5.8. Resultados

Las pruebas realizadas demuestran que el sistema implementado es capaz de realizar la detección corretamente. Se probó la aplicación dividiendo un aula en dos espacios separados aprovechando el hecho de que había dos puertas de acceso.

La colocación de los sensores es importante para el funcionamiento de la aplicación. Los sensores de movimiento se situaron en el marco de la puerta, de manera que al entrar se detectaba la presencia de forma inmediata. Los sensores de luz se distribuyeron de la misma forma que en la calibración de los mismos, uno en el centro de cada pared de la estancia, cuatro en total, con el fin de cubrir el mayor espacio posible minimizando el número de nodos. Los nodos con capacidad multimedia se situaron en una esquina enfocando al resto de la estancia y con el objetivo de que el suelo fuese visible. En la Figura 5.7 se muestra un plano del aula utilizada junto con la ubicación de los diferentes nodos.

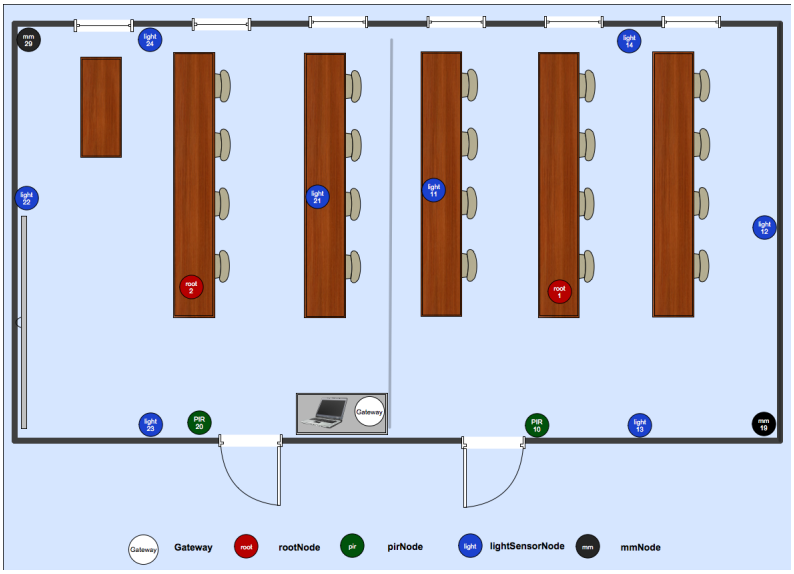


Figura 5.7: Plano del aula y ubicación de sensores para las pruebas.

Se pudo comprobar que la aplicación funcionaba como se esperaba, aunque no se consiguió un 100 % de cobertura de detección con los sensores de luz, lo que implica que sería necesario utilizar una densidad de nodos mayor. A pesar del correcto funcionamiento, se observó que se producían interferencias provocadas por la WiFi, provocando pérdida de paquetes, lo cual resulta crítico si los paquetes perdidos contienen mensajes de control.

En el Apéndice B se adjunta el fichero de salida de una de las pruebas realizadas.

Capítulo 6

Conclusiones

El objetivo de este proyecto era desarrollar una aplicación, de bajo coste y pasiva, de detección y seguimiento de personas en entornos *indoor* con capacidad de análisis contextual para detectar situaciones de peligro.

Se ha realizado un estudio de la plataforma Imote2, analizando sus posibilidades y seleccionando los sensores que podrían ser útiles para la aplicación. De este estudio se concluyó que los sensores de interés eran los de luz, movimiento, acústicos y la cámara, combinados de manera que los de luz y movimiento se utilizarían para el sistema de detección y seguimiento y los acústicos y la cámara conformarían el sistema de análisis contextual.

En una primera fase tras seleccionar los sensores se realizó un estudio del funcionamiento de cada tipo de sensor, con el fin de determinar su comportamiento y poder así fijar los parámetros necesarios para la aplicación. Así, se pudo comprobar que los sensores de movimiento resultan muy eficaces, incluso se comprobó que el rango de detección en las condiciones de interés era mayor que el especificado por el fabricante.

En cuanto a los sensores de luz, resultaron funcionar especialmente bien en condiciones de iluminación natural, condiciones en las cuales con sólo cuatro sensores distribuidos en la habitación se detectaba en el 100 % del área de interés. En condiciones de iluminación artificial, el rango de funcionamiento de cada sensor quedaba claramente reducido, de manera que no era posible detectar en todo el área sin aumentar la densidad de sensores.

Con los sensores acústicos simplemente se demostró la posibilidad de filtrar determinadas bandas de frecuencia y de utilizar la energía a corto plazo (STE) de la señal de audio para detectar picos de energía que se puedan asociar a un golpe semejante a una caída. En este apartado y para futuras ampliaciones, sería conveniente caracterizar mejor la señal producida

por una caída, de forma que su discriminación con respecto al resto de sonidos que se puedan producir a lo largo del día en una casa sea más precisa.

Finalmente, se comprobó que la cámara proporciona una resolución suficiente para distinguir un accidente en una escena. Cabe mencionar que pese a que la cámara soporta formatos a color y escalados de imagen hasta un tamaño de 40x30, no se han podido utilizar correctamente debido a problemas a la hora de configurar la cámara. Se propone como futura mejora el uso de estas características, especialmente la de escalado, ya que resultaría mucho más eficiente el envío de imágenes pequeñas y se reduciría considerablemente el tiempo de transmisión.

Para cada tipo de sensor utilizado se desarrolló una aplicación específica que permite realizar la calibración de los mismos. Estas aplicaciones pueden ser utilizadas para futuros trabajos, ya que uno de los objetivos de diseño fue la flexibilidad, permitiendo variar parámetros importantes sin necesidad de reprogramar las motas.

En cuanto a la aplicación final, se ha implementado un sistema completo de detección con un análisis contextual muy sencillo. Se ha empleado una arquitectura jerárquica en estrella, en la que existe un nodo que hace de puerta de enlace entre la red inalámbrica y el ordenador, un nodo raíz en cada área de la casa (cada estancia se considará un área diferente) y sensores dependientes de cada nodo raíz, como se mostró en la Figura 5.1. La aplicación está implementada de forma que sólo se activan los sensores correspondientes al área en la que se ha detectado la persona, permitiendo así el ahorro de energía. Se ha comprobado que la aplicación funciona correctamente y cumple con los objetivos marcados.

6.1. Problemas

La mayoría de los problemas encontrados se deben a errores hardware relacionados con las motas y a la dificultad que supone el desarrollo de las aplicaciones.

Puesto que no existe ningún simulador para Imote2, la metodología de desarrollo de aplicaciones es la siguiente: programar la funcionalidad deseada, cargar el programa y probar el funcionamiento. Si no funciona como se espera, se debe revisar el código y volver a probar. Esto supone que el desarrollo sea lento, especialmente cuando se diseña una aplicación compleja que integra nodos con funcionamiento diferente, ya que cada vez que algo no funciona es necesario reprogramar todas las motas afectadas. Además, debido a la limitación de recursos de las motas, la forma de programar debe ser muy cuidadosa para garantizar que las tareas se realizan correctamente.

El primer problema relacionado con el hardware se refiere a los sensores de luz. De acuerdo con el datasheet del sensor de luz TAOS2561 [7], éste dispone de interrupciones hardware, de manera que si el valor medido es superior al valor del registro de límite superior, o inferior al registro de límite inferior, se activa la línea de interrupción. Esto resultaría de especial utilidad en la aplicación desarrollada, puesto que todo el sistema de detección se basa en si se sobrepasan o no ciertos límites. Sin embargo, tal y como se menciona en la nota relativa a errores hardware en el manual de Imote2 [10, pg. 21], en el apartado referente a la placa ITS400, falta una resistencia de pull-up sin la cual las interrupciones del sensor de luz no tienen efecto. La nota dice lo siguiente:

> *U2(TSL2561) LIGHT_INT signal requires a 10Kohm pull up to 3V. It's missing in the current revision.*

Este problema, aunque hace que la aplicación sea menos eficiente, no es crítico puesto que se puede resolver programando adecuadamente el sistema de detección.

Por otra parte, se pudo comprobar que los sensores de movimiento no funcionaban cuando se utilizaban las placas de baterías, de manera que era necesario tenerlos conectados al ordenador. El problema en este caso es que sobra una resistencia en la placa multimedia.

A nivel de programación, fue necesario cuidar la forma de programar, especialmente los bucles, ya que debido a la limitación de recursos, en función del orden en que se colocasen las condiciones el programa podía no funcionar correctamente.

Otro aspecto que es necesario comentar se refiere al sincronismo de las motas, importante para poder reducir el ciclo de trabajo de la radio con el fin de ahorrar energía. En estos momentos es posible utilizar la radio con un 10 % de ciclo de trabajo, reducirlo más hace que la aplicación deje de funcionar correctamente. Resolver este problema abarcaría otro proyecto, por lo que se ha obviado el mismo y se deja como ampliación del presente Proyecto Fin de Carrera.

También es importante resaltar el hecho de que se emite en la misma banda en la que lo hacen Wi-Fi y Bluetooth, 2.4 GHz, por lo que se pueden producir interferencias que provoquen el mal funcionamiento del sistema, ya que la comunicación implementada no es fiable, de manera que es posible perder paquetes sin que éstos se reenvíen.

Por último, cabe mencionar que pese a la viabilidad tecnológica de la aplicación, económicamente no resultaría viable debido al precio actual de las motas. Como ejemplo, una casa con dos habitaciones, un baño, cocina y salón, conectado todo por un pasillo, harían un total de seis áreas diferentes, usando un mínimo de cuatro sensores de luz por área, sería necesario lo siguiente:

- 1 *gateway*.

- 6 nodos raíz (*rootNode*), uno por cada área.
- 6 nodos con sensor de movimiento (*pirSensor*).
- 6 nodos con capacidad multimedia (*audioCameraSensor*).
- 4 nodos con sensores de luz (*lightSensor*) por cada área, en total 30.

Cada nodo necesitará el procesador, la placa de baterías (salvo el *gateway*, que necesitará la placa interfaz IIB2400) y la placa de sensores si es necesaria. Con esto, el presupuesto sólo de coste de material asociado a las motas sería el recogido en la Tabla 6.1.

Tabla 6.1: Presupuesto para una casa.

Concepto	**Unidades**	**Coste/unidad (€)**	**Coste Total (€)**
Procesador (IPR2400)	49	145	7.105
Placa básica sensores (ITS400)	30	205	6.150
Placa multimedia (IMB400)	6	245	1.470
Placa interfaz (IIB2400)	1	145	145
Placa baterías (IBB2400)	48	100	4.800
Cable USB-miniUSB	1	2.50	2.50
Pilas AAA	144	57,46	57,46
TOTAL			**19.729,96**

El coste de material asciende a 19.729,96 €, lo que por si mismo demuestra que económicamente no es viable.

6.2. Líneas Futuras

Como se ha ido mencionando a lo largo del documento, existen varias ampliaciones que se pueden realizar a partir del trabajo realizado.

- Mejora del sistema de análisis contextual, en especial en lo referente a los sensores acústicos. Además se propone utilizar la capacidad de escalado de la cámara para reducir el tiempo de transmisión de las imágenes.
- Desarrollo e implementación de un sistema de seguimiento basado en la información binaria proporcionada por los sensores de luz.

- Mejora del sincronismo con el fin de reducir el ciclo de trabajo de la radio para ahorrar energía.
- Implementación de un sistema de comunicación fiable, especialmente para evitar la pérdida de mensajes de control.

APÉNDICES

APÉNDICE A

Presupuesto

Se presenta un estudio de la viabilidad de un sistema de detección y seguimiento de personas en entornos *indoor* con capacidad de análisis contextual para detectar posibles caídas o accidentes. A continuación se muestra el presupuesto detallado para el desarrollo del trabajo propuesto.

En la Figura A.1 se muestra el Diagrama de Gantt del proyecto. En la Tabla A.1 se resumen las fases del proyecto y el tiempo aproximado necesario para cada una de ellas. En total, el tiempo dedicado es de 2272 horas. El coste por hora del ingeniero es de 24,50 € por lo que el coste de personal asciende a 55.664 €.

Tabla A.1: Fases del Proyecto

Fase 1	Documentación y aprendizaje del lenguaje de programación y el entorno.	320 *horas*
Fase 2	Puesta en funcionamiento de nuevos sensores.	400 *horas*
Fase 3	Calibración de sensores (desarrollo de aplicaciones y toma de medidas).	272 *horas*
Fase 4	Desarrollo de la aplicación.	860 *horas*
Fase 5	Redacción de la memoria.	420 *horas*

Los costes de material se pueden desglosar en aquellos imputables al hardware de la red de sensores inalámbrica (Tabla A.2) y aquellos derivados de la programación y procesado de los datos (Tabla A.3), para lo cual se requiere un ordenador para programar la aplicación y analizar los datos, un ordenador portatil para poder realizar las calibraciones de los sensores y Matlab para el procesado de datos.

Tabla A.2: Costes asociados a las motas

Concepto	**Unidades**	**Coste/unidad (€)**	**Coste Total (€)**
Procesador (IPR2400)	31	145	4.495
Placa básica sensores (ITS400)	30	205	6.150
Placa multimedia (IMB400)	5	245	1.125
Placa interfaz (IIB2400)	4	145	580
Placa baterías (IBB2400)	33	100	3.300
Programador ARM-USB + JTAG	1	51.95	51.95
Cable USB-miniUSB	4	2.50	10
Pilas AAA	200	0.399	79,80
TOTAL			**15.791,75**

Tabla A.3: Costes Generales de Material

Concepto	**Coste Total (€)**
Ordenador gama media	1300
Ordenador portatil gama media	890
Licencia Matlab	500
Statistics toolbox	200
Curve fitting toolbox	200
Signal processing toolbox	200
Image processing toolbox	200
TOTAL	**3.090**

A partir de estos datos, el presupuesto total del proyecto queda resumido en la Tabla A.4.

Tabla A.4: Presupuesto

Concepto	**Importe**
Costes de personal	55.664 €
Coste de las motas	15.791,75 €
Costes generales de material	3.090 €
Base Imponible	74.545,75 €
I.V.A (18 %)	13.418,24 €
TOTAL	**87.963,99** €

El presupuesto total de este proyecto asciende a la cantidad de **87.963,99 €** (ochenta y siete mil novecientos sesenta y tres con noventa y nueve euros).

Leganés a 28 de febrero de 2011

El ingeniero proyectista

Fdo. Aurora Mourelle Aguado

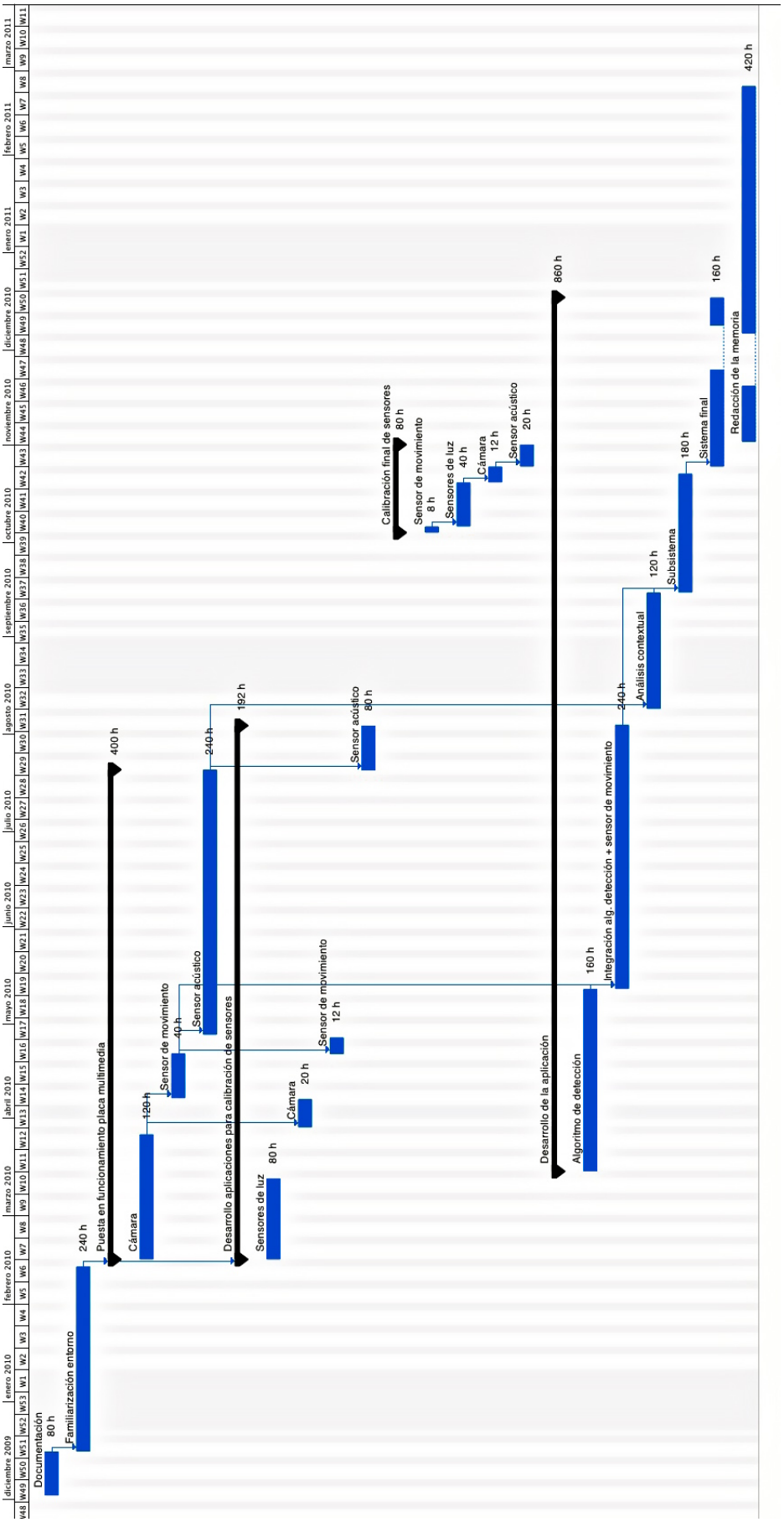


Figura A.1: Diagrama de Gantt

APÉNDICE B

Ejemplo de Fichero de Salida

A continuación se incluye un ejemplo de fichero de salida de la aplicación, creado durante las pruebas finales realizadas. Tal y como se comentó en el Capítulo 5, en el apartado de resultados, las pruebas se realizaron en un aula con dos puertas de acceso, dividiendo el espacio interior en dos zonas distintas.

En el fichero se muestran los siguientes datos:

Root: Identificador del nodo raíz responsable del sensor que envía la información.

ID: Identificador del nodo que envía la información.

BB: Valor medido por el sensor de luz.

Detect: Resultado del algoritmo de detección.

- 1: Presencia detectada.
- 2: Luces apagadas.
- 3: Luces encendidas.

State: Sentido del movimiento detectado.

- 0: Desconocido.
- 1: Acercándose al sensor.
- 2: Alejándose del sensor.

THR: Umbral de detección. Si el valor medido (BB) está por debajo de este nivel, se considera que se ha detectado presencia.

LightsOn: Umbral de luces encendidas. Si el valor medido supera este nivel se considera que se han encendido las luces.

LightsOn: Umbral de luces apagadas. Si el valor medido está por debajo de este nivel se considera que se han apagado las luces.

Power: Potencia de la señal recibida. Medida en el nodo raíz al recibir un paquete de un nodo sensor.

Date: Fecha y hora en la que se produce el evento. Es importante mencionar que este dato no lo envía la red de sensores, sino que es el ordenador el que lo fija al recibir el paquete. Se tomó esta solución debido a los problemas de sincronismo de las motas y al hecho de que el retardo de transmisión es muy bajo, ya que no se reenvían paquetes.

Para hacer más sencilla la interpretación de los datos, se va a comentar la sección correspondiente a la primera vez que se entra en el área 2 y que se muestra a continuación para mayor claridad.

Entrada en área 2 a las 22/12/2010 12:21:33

2	22	414	1	0	424	416	763	85	-60	22/12/2010 12:21:38;
2	22	400	1	0	424	416	763	85	-62	22/12/2010 12:21:38;
2	23	332	1	0	340	334	612	68	-53	22/12/2010 12:21:48;
2	23	329	1	0	340	334	612	68	-53	22/12/2010 12:21:49;
2	23	320	1	1	340	334	612	68	-52	22/12/2010 12:21:50;
2	23	295	1	1	340	334	612	68	-53	22/12/2010 12:21:51;
2	23	290	1	1	340	334	612	68	-55	22/12/2010 12:21:51;
2	23	324	1	2	340	334	612	68	-60	22/12/2010 12:21:54;
2	23	326	1	2	340	334	612	68	-55	22/12/2010 12:21:55;
2	23	327	1	2	340	334	612	68	-55	22/12/2010 12:21:55;
2	23	315	1	1	340	334	612	68	-54	22/12/2010 12:21:56;
2	23	306	1	1	340	334	612	68	-54	22/12/2010 12:21:56;
2	23	298	1	1	340	334	612	68	-51	22/12/2010 12:21:57;
2	23	300	1	1	340	334	612	68	-53	22/12/2010 12:21:58;
2	23	300	1	2	340	334	612	68	-50	22/12/2010 12:21:59;
2	23	315	1	2	340	334	612	68	-49	22/12/2010 12:21:59;

En primer lugar el *nodo 22* detecta presencia en dos medidas consecutivas y deja de detectar. El *nodo 23* detecta presencia 10 segundos después, lo cual refleja que no existe cobertura de detección en toda la estancia. Las dos primeras medidas del *nodo 23* indican que el movimiento es desconocido, y apartir de ese instante se resuelve que la persona se está acercando al sensor para luego empezar a alejarse cinco segundos después. Tres segundos después la persona cambia de sentido y de nuevo se acerca al sensor para finalmente volver a alejarse.

22/12/2010 12:21

Parámetros: T = 300ms Alpha = 2 Beta = 80

ROOT	ID	BB	DETEC	STATE	MEDIA	THR	LIGHTS ON	LIGHTS OFF	POWER	DATE
		Entrada en área 1 a las 22/12/2010 12:21:01								
1	13	635	1	0	651	638	1171	131	-64	22/12/2010 12:21:06;
1	13	633	1	0	651	638	1171	131	-62	22/12/2010 12:21:07;
1	13	628	1	1	651	638	1171	131	-65	22/12/2010 12:21:07;
1	13	585	1	1	651	638	1171	131	-70	22/12/2010 12:21:08;
1	11	390	1	0	397	390	714	80	-52	22/12/2010 12:21:08;
1	13	502	1	1	651	638	1171	131	-58	22/12/2010 12:21:08;
1	13	518	1	1	651	638	1171	131	-56	22/12/2010 12:21:09;
1	11	385	1	0	397	390	714	80	-52	22/12/2010 12:21:09;
1	11	382	1	0	397	390	714	80	-53	22/12/2010 12:21:10;
1	11	380	1	1	397	390	714	80	-59	22/12/2010 12:21:10;
1	11	385	1	2	397	390	714	80	-48	22/12/2010 12:21:11;
1	11	388	1	2	397	390	714	80	-53	22/12/2010 12:21:11;
1	11	380	1	1	397	390	714	80	-49	22/12/2010 12:21:12;
1	11	363	1	1	397	390	714	80	-50	22/12/2010 12:21:12;
1	11	379	1	1	397	390	714	80	-49	22/12/2010 12:21:13;
1	13	635	1	0	649	637	1168	130	-58	22/12/2010 12:21:13;
1	13	637	1	0	649	637	1168	130	-61	22/12/2010 12:21:13;
1	13	624	1	1	649	637	1168	130	-60	22/12/2010 12:21:14;
1	13	635	1	0	649	637	1168	130	-54	22/12/2010 12:21:15;
1	13	629	1	0	649	637	1168	130	-54	22/12/2010 12:21:15;
1	11	364	1	0	396	389	712	80	-57	22/12/2010 12:21:16;
1	11	385	1	2	396	389	712	80	-57	22/12/2010 12:21:16;
1	11	377	1	1	396	389	712	80	-55	22/12/2010 12:21:17;
1	11	382	1	1	396	389	712	80	-49	22/12/2010 12:21:17;
1	13	630	1	0	648	636	1166	130	-60	22/12/2010 12:21:17;
1	11	385	1	2	396	389	712	80	-50	22/12/2010 12:21:18;
1	13	596	1	0	648	636	1166	130	-59	22/12/2010 12:21:18;
1	13	564	1	1	648	636	1166	130	-63	22/12/2010 12:21:18;
1	13	577	1	1	648	636	1166	130	-63	22/12/2010 12:21:19;
1	13	596	1	2	648	636	1166	130	-63	22/12/2010 12:21:19;
1	13	606	1	2	648	636	1166	130	-59	22/12/2010 12:21:20;
1	13	604	1	2	648	636	1166	130	-57	22/12/2010 12:21:20;
1	14	636	1	0	675	662	1215	135	-62	22/12/2010 12:21:21;
1	13	614	1	2	648	636	1166	130	-54	22/12/2010 12:21:21;
1	12	458	1	0	468	459	842	94	-60	22/12/2010 12:21:21;
1	14	516	1	0	675	662	1215	135	-54	22/12/2010 12:21:21;
1	12	408	1	1	468	459	842	94	-57	22/12/2010 12:21:21;
1	13	628	1	2	648	636	1166	130	-60	22/12/2010 12:21:21;
1	14	512	1	1	675	662	1215	135	-60	22/12/2010 12:21:22;
1	12	427	1	1	468	459	842	94	-58	22/12/2010 12:21:22;
1	13	630	1	2	648	636	1166	130	-60	22/12/2010 12:21:22;
1	14	526	1	2	675	662	1215	135	-53	22/12/2010 12:21:22;
1	13	616	1	1	648	636	1166	130	-58	22/12/2010 12:21:22;
1	14	464	1	1	675	662	1215	135	-49	22/12/2010 12:21:23;
1	13	590	1	1	648	636	1166	130	-56	22/12/2010 12:21:23;
1	14	575	1	2	675	662	1215	135	-47	22/12/2010 12:21:23;
1	13	607	1	1	648	636	1166	130	-60	22/12/2010 12:21:23;
1	12	442	1	2	468	459	842	94	-61	22/12/2010 12:21:23;
1	14	582	1	2	675	662	1215	135	-45	22/12/2010 12:21:24;
1	12	423	1	1	468	459	842	94	-56	22/12/2010 12:21:24;
1	14	585	1	2	675	662	1215	135	-47	22/12/2010 12:21:24;
1	12	384	1	1	468	459	842	94	-55	22/12/2010 12:21:24;
1	14	579	1	1	675	662	1215	135	-45	22/12/2010 12:21:25;
1	12	365	1	1	468	459	842	94	-61	22/12/2010 12:21:25;
1	14	583	1	1	675	662	1215	135	-46	22/12/2010 12:21:25;
1	12	384	1	1	468	459	842	94	-62	22/12/2010 12:21:25;
1	14	590	1	2	675	662	1215	135	-47	22/12/2010 12:21:26;
1	13	632	1	0	647	635	1164	130	-63	22/12/2010 12:21:26;
1	12	431	1	2	468	459	842	94	-67	22/12/2010 12:21:26;

1	14	594	1	2	675	662	1215	135	-45	22/12/2010 12:21:26;
1	13	609	1	0	647	635	1164	130	-60	22/12/2010 12:21:26;
1	12	456	1	2	468	459	842	94	-69	22/12/2010 12:21:26;
1	14	591	1	2	675	662	1215	135	-46	22/12/2010 12:21:27;
1	13	607	1	1	647	635	1164	130	-60	22/12/2010 12:21:27;
1	12	458	1	2	468	459	842	94	-75	22/12/2010 12:21:27;
1	14	590	1	1	675	662	1215	135	-55	22/12/2010 12:21:27;
1	13	584	1	1	647	635	1164	130	-65	22/12/2010 12:21:27;
1	14	592	1	2	675	662	1215	135	-43	22/12/2010 12:21:28;
1	13	588	1	1	647	635	1164	130	-58	22/12/2010 12:21:28;
1	14	586	1	1	675	662	1215	135	-49	22/12/2010 12:21:28;
1	13	605	1	2	647	635	1164	130	-62	22/12/2010 12:21:28;
1	14	579	1	1	675	662	1215	135	-45	22/12/2010 12:21:29;
1	13	610	1	2	647	635	1164	130	-60	22/12/2010 12:21:29;
1	14	590	1	2	675	662	1215	135	-46	22/12/2010 12:21:29;
1	13	627	1	2	647	635	1164	130	-59	22/12/2010 12:21:29;
1	14	597	1	2	675	662	1215	135	-45	22/12/2010 12:21:30;
1	14	598	1	2	675	662	1215	135	-45	22/12/2010 12:21:30;
1	14	601	1	2	675	662	1215	135	-45	22/12/2010 12:21:31;
1	14	602	1	2	675	662	1215	135	-46	22/12/2010 12:21:31;
1	14	602	1	2	675	662	1215	135	-46	22/12/2010 12:21:32;
1	14	600	1	1	675	662	1215	135	-47	22/12/2010 12:21:32;
1	14	602	1	1	675	662	1215	135	-46	22/12/2010 12:21:33;
1	14	603	1	2	675	662	1215	135	-47	22/12/2010 12:21:33;

Entrada en área 2 a las 22/12/2010 12:21:33

2	22	414	1	0	424	416	763	85	-60	22/12/2010 12:21:38;
2	22	400	1	0	424	416	763	85	-62	22/12/2010 12:21:38;
2	23	332	1	0	340	334	612	68	-53	22/12/2010 12:21:48;
2	23	329	1	0	340	334	612	68	-53	22/12/2010 12:21:49;
2	23	320	1	1	340	334	612	68	-52	22/12/2010 12:21:50;
2	23	295	1	1	340	334	612	68	-53	22/12/2010 12:21:51;
2	23	290	1	1	340	334	612	68	-55	22/12/2010 12:21:51;
2	23	324	1	2	340	334	612	68	-60	22/12/2010 12:21:54;
2	23	326	1	2	340	334	612	68	-55	22/12/2010 12:21:55;
2	23	327	1	2	340	334	612	68	-55	22/12/2010 12:21:55;
2	23	315	1	1	340	334	612	68	-54	22/12/2010 12:21:56;
2	23	306	1	1	340	334	612	68	-54	22/12/2010 12:21:56;
2	23	298	1	1	340	334	612	68	-51	22/12/2010 12:21:57;
2	23	300	1	1	340	334	612	68	-53	22/12/2010 12:21:58;
2	23	300	1	2	340	334	612	68	-50	22/12/2010 12:21:59;
2	23	315	1	2	340	334	612	68	-49	22/12/2010 12:21:59;

Entrada en área 1 a las 22/12/2010 12:22:08

1	13	621	1	0	660	647	1188	132	-60	22/12/2010 12:22:11;
1	13	581	1	0	660	647	1188	132	-74	22/12/2010 12:22:12;
1	13	569	1	1	660	647	1188	132	-60	22/12/2010 12:22:12;
1	13	533	1	1	660	647	1188	132	-58	22/12/2010 12:22:13;
1	13	542	1	1	660	647	1188	132	-54	22/12/2010 12:22:13;
1	13	556	1	2	660	647	1188	132	-58	22/12/2010 12:22:14;
1	13	599	1	2	660	647	1188	132	-59	22/12/2010 12:22:14;
1	13	631	1	2	660	647	1188	132	-60	22/12/2010 12:22:15;
1	13	647	1	2	660	647	1188	132	-56	22/12/2010 12:22:15;

Entrada en área 2 a las 22/12/2010 12:22:26

2	23	289	1	0	338	332	608	68	-53	22/12/2010 12:22:29;
2	23	323	1	2	338	332	608	68	-58	22/12/2010 12:22:29;
2	23	323	1	2	338	332	608	68	-53	22/12/2010 12:22:30;
2	23	326	1	2	338	332	608	68	-52	22/12/2010 12:22:30;
2	23	329	1	2	338	332	608	68	-52	22/12/2010 12:22:31;
2	23	330	1	2	338	332	608	68	-51	22/12/2010 12:22:31;
2	23	332	1	2	338	332	608	68	-53	22/12/2010 12:22:32;
2	22	413	1	0	421	413	757	85	-56	22/12/2010 12:22:39;
2	22	413	1	0	421	413	757	85	-59	22/12/2010 12:22:39;

++

APÉNDICE C

Manual de Usuario

En este apéndice se detallan los diferentes ficheros que componen la aplicación final y se proporcionan las instrucciones para su correcta instalación.

La aplicación se ha segmentado en diferentes carpetas que representan cada uno de los elementos del sistema.

interfaceNode, rootNode, lightSensorNode, pirNode y mmNode: Cada una contiene el código nesC que implementa la funcionalidad del nodo con el mismo nombre. En cada una de las carpetas se encuentran los siguientes archivos:

- Makefile
- *nomTipoNodo*M.nc: Archivo nesC que contiene la implementación del componente (Módulo).
- *nomTipoNodo*C.nc: Archivo nesC que contiene la configuración del componente (Configuración).

lib: Carpeta en la que se agrupan los archivos que serán necesarios para el correcto funcionamiento de la aplicación. Las interfaces y drivers para la cámara, PIR y audio fueron desarrollados en otras universidades y distribuidos a través de grupos de contribución a TinyOS.

Camera/xbow_cb: Interfaz para el uso de la cámara.

PMIC: Interfaz para el correcto funcionamiento del sensor de movimiento. Esta interfaz permite gestionar el modo de bajo consumo y cómo salir de él. Se modificó para permitir capturar las interrupciones y gestionarlas, ya que originalmente cada vez que se generaba una interrupción sólo se podía hacer un *reset*.

WM8940: Driver para el funcionamiento del sensor acústico. Está escrito en C en lugar de nesC. Se añadieron las funciones necesarias para utilizar los filtros.

LightSensorDetection: Interfaz creada específicamente para la aplicación diseñada. Implementa el algoritmo de detección de los sensores de luz.

AudioCameraDetection: Interfaz creada específicamente para la aplicación diseñada. Implementa el algoritmo de análisis contextual.

Detection.h: Archivo con la definición de las constantes del programa y la estructura de los mensajes.

java: Carpeta con el archivo java que implementa la interfaz de usuario de la aplicación. Incluye también el Makefile.

Resultados: Carpeta en la que se almacenan automáticamente los archivos de salida de la aplicación y las imágenes que se hayan recibido.

Para instalar la aplicación es necesario utilizar la placa interfaz IIB2400, que se conecta al ordenador a través de un cable usb-miniUsb y del programador JTAG.

Se deben seguir los siguientes pasos:

1. Programar las motas[1] . Para ello es necesario abrir una consola, ir a la carpeta correspondiente al tipo de nodo que se va a programar y ejecutar el comando:

   ```
   make intelmote2 install,N
   ```

   Donde N es el identificador de la mota. En este punto es importante recordar la convención de asignación de identidades descrita en el Capítulo 5.

   - *gateway:* Siempre tendrá el identificador 0.
   - *rootNode:* Un número del 1-9. Este número identificará un área determinada.
   - Resto de nodos: El identificador será siempre el del *rootNode* correspondiente seguido de un número del 0-9. Para simplificar se recomienda utilizar el 0 para el *pirNode*, el 9 para el *mmNode* y el resto para los sensores de luz.

   Por ejemplo, en la prueba correspondiente al fichero adjunto en el Apéndice B, se utilizaron las siguientes identidades:

[1] Para programar una mota es necesario conectar la placa del procesador a la placa interfaz IIB2400, que a su vez estará conectada al ordenador a través de un cable usb-miniUsb y del programador (ARM-USB-JTAG).

- *Gateway*: 0
- Área 1:
  - *rootNode:* 1
  - *pirNode:* 10
  - *lightSensorNode:* 11, 12, 13 y 14
  - *mmNode:* 19
- Área 2:
  - *rootNode:* 2
  - *pirNode:* 20
  - *lightSensorNode:* 21, 22, 23 y 24
  - *mmNode:* 29

2. Ejecutar la GUI. En la consola, ir a la carpeta *java* y ejecutar lo siguiente, (el comando make sólo es necesario si se va a compilar de nuevo). La interfaz diseñada puede verse en la Figura 5.6.

```
[ >make clean; make]
    >java AppGUI
```

3. Fijar los parámetros de funcionamiento. Si se dejan en blanco se usarán los valores por defecto, resumidos en la Tabla C.1.

4. Pulsar *Hello* si se quiere comprobar que los nodos están funcionando correctamente.

5. Pulsar *Start* para iniciar la aplicación.

Tabla C.1: Parámetros por defecto.

Periodo Muestreo	500 ms
α	2 %
β	80 %
Ganancia	false
Frecuencia de muestreo	8 kHz
Filtro paso bajo (FPB)	false
Filtro paso alto (FPA)	false
FPA primer orden	false
Umbral de sonido	10^7
Periodo de ventana	30 ms
Imagen a color	false
Ruta archivo imagen	../Resultados/imagenes
Ruta archivo salida	../Resultados/ouput

C.1. Relación de Aplicaciones

A continuación se proporciona un listado de las aplicaciones desarrolladas y que se encuentran en el CD adjunto con este PFC.

- Aplicaciones de calibración de sensores: /PFC_AuroraMourelle/SensorCalibration/
  - *LightSensorCalibration*
  - *AudioCalibration*
  - *CameraCalibration*
  - *PIRCalibration*
- Aplicación final: /PFC_AuroraMourelle/FinalApp
  - *FinalApp*

Bibliografía

[1] http://www.imserso.es/InterPresent1/groups/imserso/documents/binario/idi3_06cystelcommad.pdf.

[2] http://www.pervaya.com/esp/index.html.

[3] http://tech.groups.yahoo.com/group/intel-mote2-community/.

[4] *Datasheet* IMB400. http://www.memsic.com/support/documentation/wireless-sensor-networks/category/7-datasheets.html?download=172%3Aism400-imote2&start=20.

[5] *Datasheet* Imote2. http://www.memsic.com/support/documentation/wireless-sensor-networks/category/7-datasheets.html?download=134%3Aimote2.

[6] *Datasheet* ITS400. http://www.memsic.com/support/documentation/wireless-sensor-networks/category/7-datasheets.html?download=137%3Aits400.

[7] *Datasheet* tsl2561. http://www.taosinc.com/downloaddetail.aspx?did=140.

[8] *Datasheet* PIR. http://panasonic-denko.co.jp/ac/e_download/control/sensor/human/catalog/bltn_eng_mp.pdf.

[9] *Datasheet* wm8940. http://www.wolfsonmicro.com/products/codecs/WM8940/.

[10] Imote2 *hardware reference*. http://www.memsic.com/support/documentation/wireless-sensor-networks/category/6-user-manuals.html?download=56%3Aimote2-hardware-reference-manual.

[11] Manual de programación de TinyOS. http://www.tinyos.net/tinyos-2.x/doc/pdf/tinyos-programming.pdf.

[12] Tutorial TinyOS. http://docs.tinyos.net/index.php/TinyOS_Tutorials.

[13] IEEE Standard 802.15.4, 2003. http://standards.ieee.org/getieee802/download/802.15.4-2003.pdf.

[14] Las personas mayores en españa, 2008. http://www.imsersomayores.csic.es/estadisticas/informemayores/informe2008/index.html.

[15] F. Bagci, F. Kluge, T. Ungerer y N. Bagherzadeh. *LocSense - An indoor location and tracking system using wireless sensors.* En *Computer Communications and Networks, ICCCN*, pp. 1–5, agosto 2008.

[16] S. Blackman y R. Popoli. *Design and analysis of modern tracking systems.* Artech House, 1999.

[17] M. D'Souza, T. Wark y M. Ros. *Wireless localisation network for patient tracking.* En *Intelligent Sensors, Sensor Networks and Information Processing, ISSNIP*, pp. 79–84, diciembre 2008.

[18] D. Gay, P. Levis y R. von Behren. *The nesC Language: A Holistic Approach to Networked Embedded Systems.* www.tinyos.net/papers/nesc.pdf.

[19] M. Johnson, M. Healy, P. van de Ven, M. J. Hayes, J. Nelson, T. Newe y E. Lewis. *A comparative review of wireless sensor network mote technologies.* En *IEEE Sensors*, pp. 1439 –1442, 2009.

[20] T. Mori, H. Noguchi, Y. Suemasu y T. Sato. *Multiple people tracking by integrating distributed floor pressure sensors and RFID system.* En *SMC (6)*, pp. 5271–5278, 2004.

[21] L. Nachman, J. Huang, J. Shahabdeen, R. Adler y R. Kling. *Imote2: Serious computation at the edge.* En *Wireless Communications and Mobile Computing Conference, IWCMC*, pp. 1118–1123, agosto 2008.

[22] J. G. Proakis y D. K. Manolakis. *Digital Signal Processing: Principles, Algorithms and Applications.* Prentice Hall, 1996.

[23] K. Sohraby, D. Minoli y T. Znati. *Wireless Sensor Networks. Technology, Protocols and Applications.* John Wiley and Sons, 2007.

[24] R. Want, A. Hopper, V. F. ao y J. Gibbons. *The Active Badge location system. ACM Transactions on Information Systems (TOIS)*, 10:91–102, 1992.

[25] F. Zhao y L. Guibas. *Wireless Sensor Networks. An information processing approach.* Elsevier, 2004.